Pedro Ricardo Contradanças de Andrade

Licenciado em Ciências de Engenharia
Electrotécnica e de Computadores

Enterprise Reference Lexicon Building from Business Models

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador: Ricardo Jardim-Gonçalves, Professor Associado com Agregação, DEE, FCT-UNL
Co-orientador: João Filipe dos Santos Sarraipa, Investigador, UNINOVA

Júri:

Presidente:	Prof. Doutor João Francisco Alves Martins
Arguente:	Prof. Doutora Teresa Cristina de Freitas Gonçalves
Vogais:	Prof. Doutor Ricardo Luís Rosa Jardim-Gonçalves

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2014

Enterprise Reference Lexicon Building from Business Models

To my parents and my beloved Lucía

Acknowledgements

The conclusion of this dissertation represents, without a doubt, the end of another very important chapter of my life, in a way that someday i will be able to look back and shed a tear or two of a very nostalgic joy. However, this goal could never be achieved without the support and love of my parents, Ana Contradanças and João Andrade, for believing in my capabilities and providing me with everything they could in the best way they knew, thank you!

I would like to thank my thesis supervisor, Professor Ricardo Gonçalves for giving me the opportunity to work with him and with his research team at GRIS.

I would like to show a deep appreciation to my thesis advisor, João Sarraipa, for his continuous help, dedication, commitment, support and especially patience during all the process of researching and developing a solution.

I would like to give a special thanks to Matthew Curland, from ORM Foundation, who was humble enough and had the patience to hear all my questions and to assist me without hesitating.

I would also like to thank all my college colleagues, especially Fábio Branco (*"Está feito páh!"*), Afonso Maria, Pedro Pires, Gonçalo Delgado, Pedro Almeida, Nuno Pereira, Henrique Chambel, Fábio Miranda and Jorge Claro, for all their support, help and for being able to share a good laugh with me.

Finally, I would like to thank my beloved girlfriend Lucía, for her unconditional love and friendship and for never having stopped believing and supporting me, even on the most distressed times where everything seemed to fail and fall apart, helping me rise and face the mountains with hope and faith.

Abstract

Nowadays, a significant increase on the demand for interoperable systems for exchanging data in business collaborative environments has been noticed. Consequently, cooperation agreements between each of the involved enterprises have been brought to light. However, due to the fact that even in a same community or domain, there is a big variety of knowledge representation not semantically coincident, which embodies the existence of interoperability problems in the enterprises information systems that need to be addressed.

Moreover, in relation to this, most organizations face other problems about their information systems, as: 1) domain knowledge not being easily accessible by all the stakeholders (even intra-enterprise); 2) domain knowledge not being represented in a standard format; 3) and even if it is available in a standard format, it is not supported by semantic annotations or described using a common and understandable lexicon.

This dissertation proposes an approach for the establishment of an enterprise reference lexicon from business models. It addresses the automation in the information models mapping for the reference lexicon construction. It aggregates a formal and conceptual representation of the business domain, with a clear definition of the used lexicon to facilitate an overall understanding by all the involved stakeholders, including non-IT personnel.

Keywords: Fact Models, Information Systems, Interoperability, Knowledge Representation.

Resumo

Hoje em dia, tem-se notado um aumento significativo na procura de sistemas que interajam através da troca de dados dentro de um ambiente colaborativo. Consequentemente, foram definidos acordos de colaboração entre cada uma das empresas envolvidas. Contudo, devido ao facto de existir uma grande variedade de representações de conhecimento entre as diversas comunidades, existem inúmeros problemas a nível de interoperabilidade que necessitam de ser ultrapassados.

Ademais, em relação a isto, a maioria das organizações enfrenta outros problemas dentro da temática de sistemas de informação, tais como: 1) O domínio das organizações não se encontrar facilmente acessível por todas as partes interessadas (mesmo dentro das próprias empresas); 2) O domínio das organizações não se encontrar representado num formato padrão; 3) E mesmo estando representado num formato padrão, não ser suportado por anotações semânticas ou descrito usando um léxico comum e compreensível.

Esta dissertação propõe uma abordagem para o estabelecimento de um léxico de referência para empresas através de modelos de negócios. É abordada a automação de mapeamento de modelos de informação para a construção de um léxico de referência. Reúne uma representação formal e conceptual do domínio de negócios, com uma clara definição do léxico usado para facilitar a compreensão por parte de todos os membros envolvidos, incluindo membros que não estejam familiarizados com os aspectos mais técnicos.

Palavras-Chave: Modelos de Factos, Sistemas de Informação, Interoperabilidade, Representação de Conhecimento.

Table of Contents

List of Figures

List of Tables

Acronyms

API – Application Programming Interface

BPMN – Business Process Model and Notation

CLR – Common Language Runtime

CNL – Constrained Natural Language

DB – Database

DL – Data Language

EI – Enterprise Interoperability

EISB – Enterprise Interoperability Science Base

ER – Entity-Relationship model

FCL – Framework Class Library

FM – Fact Model

HTML – Hypertext Markup Language

ISO – International Organization for Standardization

KB – Knowledge Base

KMType – Knowledge Mapping Type

KR – Knowledge representation

KRE – Knowledge Representation Element

LCIM – Levels of Conceptual Interoperability Model

LINQ – Language-Integrated Query

MAMP – Microsoft, Apache, MySQL, PHP/Perl/Python

MatchClass – Match/Mismatch Classification

Melems – Mapping Element Pairs

MENTOR – Methodology for Enterprise Reference Ontology Development

MO – Mediator Ontology

NL – Natural Language

NLP – Natural Language Processing

NORMA – Natural ORM Architect for Visual Studio

OCL – Object Constraint Language

OMG – Object Management Group

ORM – Object Role Modeling

OWL – Web Ontology Language

PLiX – Programming Language in XML

POS – Part-of-Speech

RDF – Resource Description Framework

RDMS – Relational Database Management System

REST – Representational State Transfer

SBVR – Semantics of Business Vocabulary and Business Rules

SME – Small-Medium Enterprises

SQL – Structured Query Language

TDM – The Decision Model

UML – Unified Modeling Language

URI – Uniform Resource Identifier

VEA – Visio for Enterprise Architects

W3C –World Wide Web Consortium

XML – Extended Markup Language

1

Introduction

In today's reality, small and medium enterprises (SME) find themselves facing an ever-growing business market. In order for those companies to keep in business and don't get swallowed by the big names on the market, they must increase their competitiveness and productivity. With the well-known market globalization, these SME have soon realised that for them to keep their pace and don't fall, they would need to cooperate with each other, allowing their costumers needs to be fulfilled with real-time response. However, there is always a price to pay when referring to enterprise collaboration.

Due to the worldwide diversity of the communities, a high number of knowledge representation elements (KRE), such as ontologies and fact models, which are not semantically coincident, have appeared representing the same segment of reality [1]. As a consequence of that, enterprises face many problems when trying to communicate with each other, since most likely none of their systems use the same type of model to represent their domain of discourse.

To solve this interoperability problem, many methodologies have been proposed, either by collaboratively building a reference knowledge representation element by using qualitative information collection methods [1], or either by agreeing on the use of a specific standard, like ISO, or even by creating a semantic interoperability check frameworks [2]. All those solutions are viable, though none ensure an automated construction of a common path to connect all the involved business information models.

Another issue that comes is the fact that the majority of the knowledge representation elements used by organizations are not attribute-free (UML, OWL and ER for instance). This causes a lot of trouble when trying to change the domain. A solution for that issue is to use Fact Models as the domain representation model, as it is attribute-free and allows the conceptual representation of the knowledge.

1.1 Context and Motivation

With the continuous evolution of technology, data exchange between enterprises has become an increasingly important goal to be reached. In order for this to happen, cooperation agreements among each of the concerned stakeholders must take place.

Given the importance of business rules, the knowledge representation elements that govern an organization should be modelled in such way that makes all the rule statements and business concepts to be easily accessible by all stakeholders, be in a standard format, and use business terminology, rather than database object names, which force the business stakeholders to either learn the used notation and possibly become familiar with the relevant database object names, or rely on the technician responsible for building the business KRE, to explain the used notation [3].

Indeed, business rules and enterprise interoperation represent one of the 21st century's major technological challenges, and if we are to create optimal information systems, then we should develop consistent business rules and methodologies.

This dissertation proposes an approach for the establishment of an enterprise reference lexicon from business models. It addresses the automation in the information models mapping for the reference lexicon construction. It aggregates a formal and conceptual representation of the business domain, with a clear definition of the used lexicon to facilitate an overall understanding by all the involved stakeholders, including non-IT personnel.

1.2 Research Method

The research method used to develop this dissertation was based on the Classical Method. This method is composed by seven steps, starting by a more theoretical point of view, finishing with the obtained practical results, as described in Fig. 1.1:

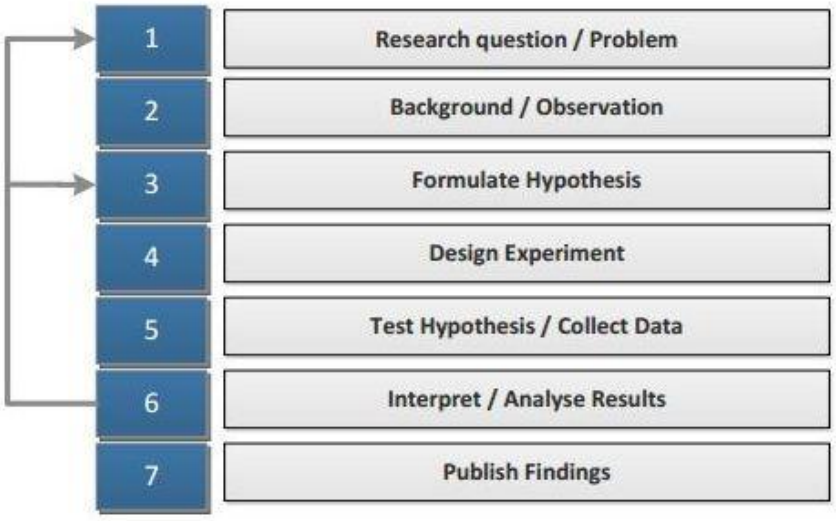


Fig. 1.1: Phases of the Classical Research Method [4]

A concise description of each of the phases presented in Fig. 1.1 is presented as follows:

1. **Research Question / Problem:** This is one of the most important steps during a research, since this is the period where the researcher must identify the problem at hand and define the research question. This research question must be clearly defined, as it will be the base for all the work. This research question is defined in section 1.3;

2. **Background / Observation:** This step aims to show the reader all the background work that was necessary to do before starting to develop a viable solution to the research question. That work includes the study of several scientific papers and publications that present solutions to similar problems. Although being important to have a varied collection of scientific documents, it is critical that the researcher takes notice to the innovation/reliability ratio, since some of the literature can be obsolete, due to its age or for not being reliable, due to low experimentation. With that said, the researcher is able to use this background study to note what innovation his work brings relatively to the previous work. The state of the art of this dissertation is presented in the sections 2 and 3;

3. **Formulate Hypothesis:** This step grasps the results that the researcher is expecting to obtain after the development of the proposed solution. This step has the main purpose of bringing clarity to the researcher research question. The hypothesis is presented in section 1.4;

4. **Design Experiment:** In this step, the researcher has to present the architecture plan for the experimental implementation, validating the hypothesis previously presented. This step is presented in section 4;

5. **Test Hypothesis / Collect Data:** This is the step where the researcher has to dedicate his time to get a practical implementation of the purposed solution. The researcher has to test the presented system architecture using distinct scenarios and then evaluate the obtained results in the next step.

6. **Interpret / Analyse Results:** After testing the system architecture in the last step, the researcher has to evaluate and analyse the obtained results. With these results, the researcher can now get to a conclusion if the hypothesis that he presented can be considered as valid or not. If so, the researcher can be granted his deserved rightfulness and he can consider the next steps that come after, making some recommendations for further research. On the order hand, if the obtained results prove that the hypothesis presented by the researcher was wrong, the researcher

should not take this has a failure, but as a chance to improve the initial approach and go back to the first phase of the research method.

As Thomas Edison once said "*I haven't failed. I've just found 10000 ways that won't work*";

7. Publish Findings: Finally, if the researcher indeed got to a set of results that prove that his hypothesis was right, he then must end up publishing his findings, contributing to the scientific community. These findings can be then presented in scientific conferences, where the author has to present his ideas for the research and the obtained results.

1.3 Research Problem and Questions

- Does the establishment of semantic mapping representations between business models support an enhanced and formal way of construction of its reference lexicon?

1.4 Hypothesis

- If a methodological approach for building a domain reference lexicon is defined based on a well-known methodology for reference ontology building as MENTOR, which accomplishes semantic mapping tables, then automatism establishment on the approach can be eased.

1.5 Dissertation Outline

This dissertation is divided in six sections. Each of the sections will comprehend:

- Section 1 – Introduction: This section reveals the purpose of this work as well as the motivation behind this research project. Then it reveals the adopted research method that the author chose. Furthermore, the research questions that motivated this dissertation are presented and finally the author presents the hypothesis that he tested in order to solve those questions.
- Section 2 – Knowledge Representation Based on Natural Language: This chapter is used to present the State of the Art of this dissertation. It represents the study conducted by the author in order to have enough information for the development of the solving methodology. It addresses the main types of business models available and presents some of the available tools in the market.

- Section 3 – Establishing an Enterprise Reference Lexicon: In this chapter, the author covers the background research about the interoperability problems and the levels of maturity necessary for a functioning interoperable system environment. Furthermore this chapter covers an existing methodology that was developed to solve this problem.
- Section 4 – Proof-of-Concept Implementation: In this chapter, the author purposes a methodology adaptation that is believe to respond to the dissertation problem and that goes according to the presented hypothesis. The architecture of the developed methodology is demonstrated and explained by the detail, indicating the used tools and technologies.
- Section 5 – Demonstrator Testing and Hypothesis Validation: This practical chapter is used to present to the reader the implementation of a prototype conducted by the author in order to prove the presented hypothesis, showing the obtained results after executing an example domain structure.
- Section 6 – Conclusion: Finally, this dissertation is finalized by presenting the final thoughts and remarks, validating, or not, the presented hypothesis for the solving of the research questions and problems identified previously. This chapter finish with a proposition for possible future work topics.

2

Knowledge Representation Based on Natural Language

This chapter provides a theoretical overview of the most essential concepts behind knowledge representation of business domains. The first section gives a brief introduction to knowledge representation and discusses the importance of using natural language for the design of formal and reliable business models, followed by a section with an introduction to some distinct knowledge representation models, such as the Ontologies, The Decision Models (TDM) and the Fact Models (FM), where in this final case, an introduction to some of the available development tools is presented. Finally, the author expresses the main differences between the studied models and discusses their capabilities and potential inside business environments.

2.1 Knowledge Representation

Knowledge Representation is a field of Artificial Intelligence (AI) that focuses on the formalization of knowledge and its processing within machines. In a wider sense, KR is concerned to representing information about the world in a form that computer systems can utilize to solve complex tasks with natural language formalism [5]. However, there is a rather tendency to separate KR from natural language formalism. The reason for this is that natural language can be very ambiguous and so, knowledge must be expressed in a way that one can easily identify the structure and the characteristics of concepts and the relationships among them [6].

A Knowledge Representation Element (KRE) is an element that aids on the representation of formalized knowledge in a specific domain [7]. The Knowledge Base is the most generic KRE and represents all the amassed enterprise knowledge. In order to get to this final knowledge representation, enterprises need to clearly define their business terminology, document its terms and respective descriptions in a Glossary and finally define the taxonomic relationships between these terms, creating a Thesaurus.

To build a proper domain knowledge base, a certain path has been defined [8], representing different levels of conceptualization. The following figure, illustrates this path:

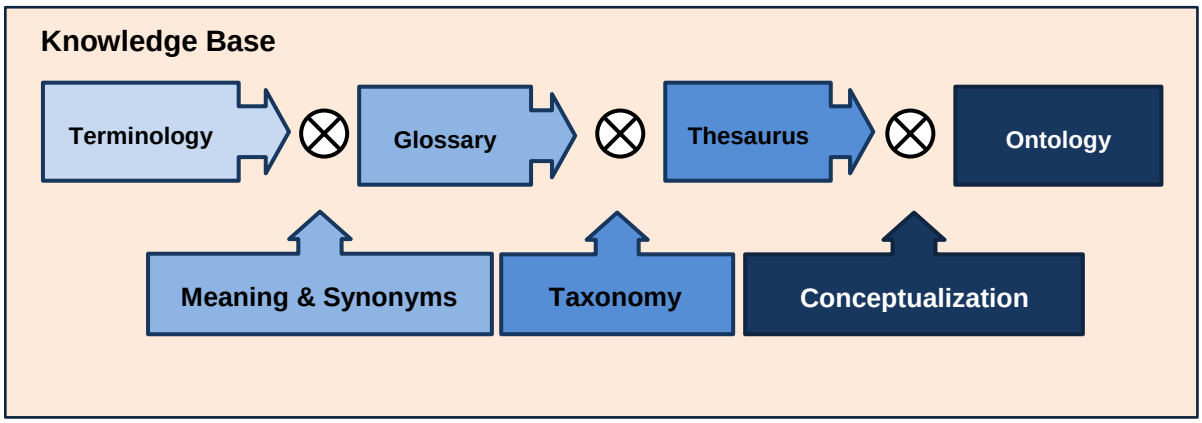


Fig. 2.1: Knowledge Representation Elements Relationship

A glossary is generally useful for unifying knowledge sharing during the development of an enterprise business Knowledge Base. It comprehends an alphabetic list of all the terms used by the organization, including their respective descriptions (meaning), synonyms and references.

When applying a taxonomic structure to the organization terms, we get a thesaurus. A thesaurus provides hierarchical relationships between the organization terminologies, allowing one to group different terms into a more general, higher category.

Finally, the obtained structure can be transformed into a conceptualized model, such as, ontologies. In the ontologies, the business concepts are described with the aid of classes, relations and attributes, using natural language to describe the meaning of these entities and formal axioms to constraint their behaviour.

Currently one of the most active areas of knowledge representation research, are projects associated with the Semantic Web, which has the objective of adding a layer of semantics on top of the Internet. The advantage of this is that, instead of tagging/indexing web sites with keywords, the Semantic Web organizes these web sites in a large ontology of concepts.

2.2 Natural Language

Due to the continuous growth across all business segments and market, the creation of methodologies to provide interoperability between enterprises has become a priority. However many problems, particularly regarding the semantics of the concepts involved have been identified during this process.

The rules governing an organization and its systems need to be documented in such a manner as to enable stakeholders to review them for relevance and correctness, approve them, change them as required, and so on. This requires in turn that all rule statements be easily accessible by all stakeholders, be in a standard format, and use business terminology (rather than database object names) [3].

Unfortunately, this is easier said than done. Current technology does not well support this requirement. Although the fact that the average rule engine is able to provide a full set of rule statements in a standard format, this format is very unlikely to be understandable to all the involved stakeholders, since it's mainly represented has database object names. This issue led to the development of tools that rely on natural language for the representation of a domain.

As stated by Fatwanto, there are two reasons why natural language is preferred as the specification mean. First, most of the involved stakeholders are more familiar with natural language compared to using other types of media, like formal or modeling language. The second reason is that specifying software requirements using natural languages is the most common practices in software development projects [9].

Natural language is defined as any language, which arises in an unpremeditated fashion as the result of the innate facility for language possessed by the human intellect. A natural language is typically used for communication, and may be spoken, signed, or written. Natural language is distinguished from constructed languages and formal languages such as computer-programming languages or the "languages" used in the study of formal logic, especially mathematical logic [10].

Naturally, the use of natural language for the representation of business concepts presents many benefits to all the stakeholders. However, natural language can be very ambiguous. This language ambiguity can be noticed either at a syntactic, semantic, structural or pragmatic level.

Hence, the English specifications of software requirements can not only result in erroneous and absurd software designs and implementations but the informal nature of English is also a main obstacle in machine processing of English specification of the software requirements. To address this key challenge, there is a need to introduce a controlled NL representation.

2.2.1 Ambiguity Levels of the Natural Language

Natural Language has some requirements in order to achieve a quality state. The main components in NL analysis are [11][12]:

- **Morphology:** The study of how words are constructed; prefixes & suffixes. For instance, the word "drive" has the prefix "drove" and the suffixes drives, driving, driven;

- **Syntax:** The study of the structural relationships between words, defining sentence formation. For example, using the sentence "The employee drives a car", we can apply the part-of-speech(POS) approach:

  " The/**DT** employee/**NN** drives/**VBZ** a/**DT** silver/**JJ** car/**NN** "

  , where **DT** stands for determiner, **NN** stands for noun, **VBZ** stands for present simple verb and **JJ** stands for adjective;

- **Semantic:** The study of the meaning of words, phrases, and expressions. It is possible to have the same noun to express different objects, showing up lexical ambiguity. For instance:

  "A car is a motorized vehicle" or
  "A car is a vehicle hauled by a locomotive"

  Has we are able to observe, this can be a real issue, as it shows up lexical ambiguity for the same word "car";

- **Discourse:** The study of the contextual effects and the relationships across different sentences or thoughts, as in:

  "**CoTech** was founded at 2010" or
  "**Contradanças Technologies** will be releasing a new product"

- **Pragmatic:** The study of the purpose of each statement and the usage we give to language in specific contexts, like concluding the way we should react to a certain sentence, giving the right response on the right situation.

All this NL components show up how ambiguous the language can be, making the processing of this language a very hard task. While some of these knowledge aspects can be memorized like the suffix "talking" of the word "talk", as "talk+ing", other words can't benefit of that aspect, like in the word "drive" that has the suffix "driving". In this case, the machine needs to know that the word "driv" isn't really a word.

A potential solution would be to use a probabilistic model built from language data, indicating how similar two words might be, for instance [12]:

P ("Car" → "Vehicle") **Medium Similarity**
P ("Car" → "Automobile") **High Similarity**

There are already many examples of Natural Language Processing approaches for deriving formal specifications from informal ones, such as, Conceptual Model Builder[13], LIDA [14], NLP [15], and many more.

2.2.2 Constrained Natural Language

A Constrained Natural Language (CNL) is a subset of NL that has been restricted with respect to its grammar and its lexicon [16].

By restricting the grammar, sentence structures can be simplified and standardized. By restricting the lexicon, unnecessary linguistic variations can be removed, and retained words can be less ambiguously defined. This improves the semantic quality of a specification [11], allowing humans to read and understand texts with more ease and at the same time allowing machines to process that data more effectively.

To attend this, Schwitter developed Processable ENGlish [16]. PENG is a computer-processable controlled natural language designed for writing unambiguous and precise specifications, that is, specifications that only have one possible interpretation. PENG covers a strict subset of Standard English and is precisely defined by a controlled grammar and a controlled lexicon [5].

As a consequence of constraining the language, some disadvantages may occur. By constraining the language, there is a possibility, and for some, an unavoidable consequence (as in [17]) of reducing the expressiveness. This may happen because the expressiveness of the language is a measure of the variety of lexical and grammatical constructions it allows [17].

A good example of the use of a constrained NL, is the specification method used in [9]: The specification method used starts by asking all the stakeholders for their concerns in relation to the system under development. The answers are then analyzed and contested and if there is any conflict, the stakeholders need to come to an agreement. Once the stakeholders come to an agreement towards all the conflicting concerns, the specification method can then derive the concerns into requirements of the system. The format used to specify the requirements is:

$$\text{Requirement} \leftarrow \text{Subject} + \text{Verb} + \text{Target} + [\text{Way}]$$

,where a requirement represents an action performed by an agent who affects one state of an entity/object, subject represents the agent, verb represents the activity taken by the agent, target represents the object and way defines the way in which an action will be taken.

2.3 Semantics of Business Vocabulary and Business Rules

In June 2003 the Object Management Group (OMG) issued the Business Semantics of Business Rule (BSBR) Request For Proposal in order to create a standard to allow business stakeholders to define the policies and rules by which they run their business in their own language, in terms of the things they deal with in business, and to capture those rules in a way that is clear, unambiguous and readily translatable into other representations [18].

The Semantics of Business Vocabulary and Business Rules (SBVR) is the adopted standard introduced by OMG, intended to be the basis for formal and detailed natural language declarative description of a business. SBVR is intended to formalize complex compliance rules, such as operational rules for an enterprise, security policy, standard compliance, or regulatory compliance rules [13]. Such formal vocabularies and rules can be machine-processed.

A SBVR rule is the key constituent of SBVR standard. A SBVR rule can easily be machine processed to perform object rule modeling, perform rule consistency analysis, or generate formal representations such as Object Constraint Language (OCL) constraints, databases, business rules repositories, business blueprints, business object models, software components, etc. [20]

The SBVR vocabulary allows one to formally specify representations of concepts, definitions, facts, and rules of any knowledge domain in natural language [21]. These features make SBVR well suited for describing business domains and requirements for business processes and information systems to implement business models. The following figure shows an overview of the SBVR Metamodel:

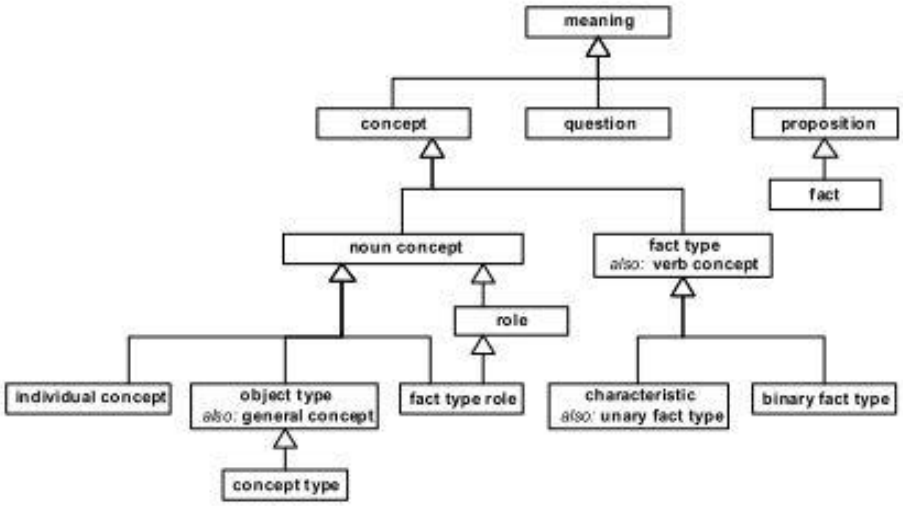
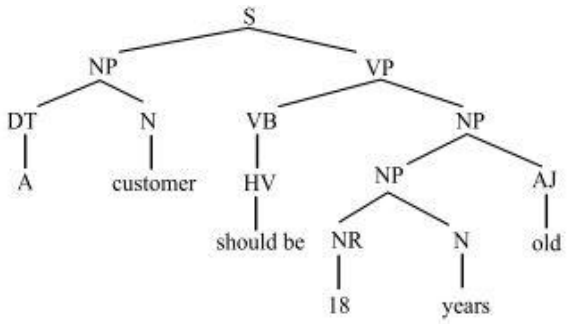


Fig. 2.2: SBVR Metamodel [19]

Bajwa and Lee developed NL2SBVR, a NL-based approach for generating SBVR business rules from English text with respect to a target business domain [21]. The tool gets the text document with the business constraints and a UML class model that provides a business domain, as input. The text is then syntactically and semantically analysed by a NL processing module. Fig. 2.3 illustrates the result of the tokenization and Parts-of-Speech (POS) Tagging of the sentence "A customer should be 18 years old":



S → Sentence; V → Action Verb;
NP → Noun Phrase; DT → Determiner;
VP → Verb Phrase; AJ → Adjective;
VB → Verb; N → Noun;
P → Pronoun; NR → Number;
HV → Helping Verb; PP → Preposition;

Fig. 2.3: Parse Tree [22]

The results generated by the NLP module are then mapped to the input UML model to assure that the obtained SBVR rules are related to the UML model. The SBVR elements are then extracted from the output of the NLP module (e.g. noun concept, object type, individual concept, verb concept, etc). With this information is then possible to create a SBVR business rule.

2.3.1 Business Rules

The definition of the "business rule" term has suffered many mutations along time, as some of the purposed definitions have conflicted with others.

According to [23], a business rule is a statement that defines or constrains some aspect of the business. It is intended to assert the business structure or to control or influence the behavior of the business.

On the other hand, Ronald Ross at [24], defined a business rule as a rule that is under business jurisdiction. Later this definition has been extended by OMG, declaring that the rules could be changed or discarded by the semantic community. Admittedly, the SBVR included the definition "…*a law or principle that operates within a particular sphere of knowledge, describing, or prescribing what is possible or allowable*" [19].

Any business, indeed any organization, is governed by a wide variety of rules. While many of these have been established by the organization itself and will therefore be able to be derived and modified by the organization own goals and purpose (Internal Rules), others will reflect legislation or regulation, external standards or best practice, or even, laws of physics (i.e. a person cannot be placed in two different locations at the same time). These can be determined as External Rules [3][25].

Witt refers on [3] that, an organization's rules should be managed in such a way that all involved entities (employees, customers, suppliers, business partners), who have a stake in those rules being complied with, or who are tasked with implementing those rules, know what rules are in force and which rules apply to each situation they may encounter.

To achieve those conditions, the organization's rules need to be managed according to the following conditions [3]:

- Each rule should be documented in one or more of the natural languages (not programming languages).
- Each natural language rule statement uses business terminology rather than database table or column names or program module names.
- Each rule statement should be unambiguous, having only one, obvious interpretation [26].
- Each natural language rule statement should be succinct (uses no more words than necessary).
- All natural language rule statements should be consistent in terms of vocabulary (terms used) and syntax (sentence structure), not contradicting one another [26].
- Each business rule needs to be atomic, that is, they cannot be broken down further, belonging to exactly one category [25].

2.3.1.1 Putting Business Rules into Perspective

To better comprehend the role of business rules inside a business, the Fig 2.4 has been presented to show the relations between these concepts:

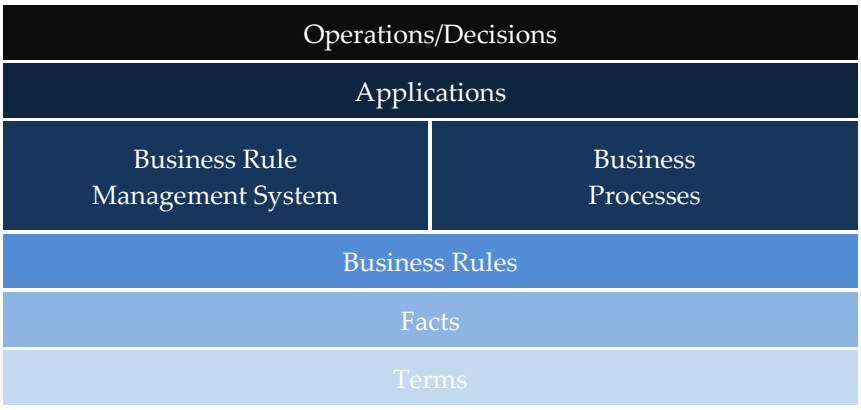


Table 2.1: Business Rules in Perspective

In order to avoid ambiguities and provide consistency for the whole business, all the concepts related to the business domain need to be properly defined. These standardized concepts are named as terms.

All the terms and their definitions are comprised in a glossary, commonly named as concept catalog [27].

By using the following rule, "A guitar is played by at most a person" it is possible to notice that the term *guitar* relates to another term *person.* These relations between terms are called fact types [27]. Fact types do not define any constraints for these relations, but rather a general connection between terms [26], so a fact type for this rule would be "guitar is played by person".

In a Business, all the terms and fact types are comprised in a fact model. The fact model describes all the business knowledge and consequently provides a common language that can be used by all business workers, being represented either textually or visually [27].

As it was already described, business rules are statements that constrain the facts present in the fact type. Having the business rules defined, these need to be stored in a rule repository allowing that every rule statement is easily accessible by all stakeholders. Finally, the operations and decisions take all parts into account [28].

2.4 Business Models

This section provides an overview of three recent business models, two for data representation and another for decision support, very useful when using it in higher level business models like the Business Process Model and Notation (BPMN). Moreover, an introduction to the ORM and OWL language is presented.

2.4.1 The Fact Model

A Fact Model (also known as a structured business vocabulary) is a business-oriented representation of the domain key concepts and the facts that relate them [29] .

The Fact Model approach has the main goal of structuring the essential knowledge about business operations in a more user-friendly way, using standard vocabulary and thus allowing the expression of each element of the business terminology in a design-independent fashion, making it possible for non-IT stakeholders to participate more actively on the development.

This type of model focuses on logical basic connections (called Fact Types) between core concepts (represented by Terms) of the business in an intuitive manner. In this sense, a Fact Model is the necessary starting point for developing the next steps of business knowledge, as constraints and rules.

In this thesis, every example of fact model content will be described in the following specification:

- Terms will be underlined;
- Verbs and prepositions will be in **bold**;
- Proper names will be double underlined.
- Constraints will be in *italic*

In order to understand the difference between a fact type and a rule in a more clear way, if we were to pick for instance, the fact type "customer places order" and then extend it to "customer **places** *at least one* order", then we would be adding a constraint to the fact type and thus, creating a rule.

Besides the fact that businesses are run on rules, fact types bring lots of benefits, making the visualization of all the business concepts very intuitive, even for non-IT personal.

2.4.1.1 Properties of the Fact Models

A Fact Model consists of:

- Concepts that all the employees, customers, suppliers and partners need to understand and refer to;
- Terminology, used to refer to those concepts and fact types;
- Fact Types, that represent the relationships between the involved concepts, each of which:
  a) Relates two or more concepts (as terms), or
  b) Documents a characteristic of a concept.

The Concept Model is one of the earliest examples of fact oriented model use, which uses the SBVR vocabulary as a standard, and was released with SBVR version 1.1.

2.4.1.1.1 Business Terms

According to [30], a term is a basic word or phrase that's used to describe or express one or more instances of a concept and that in this case has some specific, defined, unique business meaning. A collection of terms about a business constitutes a structured business vocabulary.

In Fact Models, terms can be expressed as a simple noun or a compound noun. Compound nouns are made up of simple nouns, adjectives, prepositions, numbers, conjunctions, or even verbs. However, Compound nouns can create trouble, for instance [31]:

- A noun suffixed by other noun ("Employee", "Employee Number");
- A noun prefixed by an adjective and/or other nouns ("Car", "Company Car");
- A noun suffixed by a preposition and a noun ("date of birth", "birth date").

Terms can be classified as [3]:

- Intentional: The objects covered by the business term are defined as members of a more general class with one or more distinguishing characteristics. For example the term “minor” can be used to describe a “person” whose “age” is less than “age of majority”;
- Extensional: The objects covered by the business term are listed. For example, “minor”: a “boy” or a “girl”.

2.4.1.1.2 Fact Types

Fact types are used to specify the relationship among different concepts in business rules. Fact types are formed by terms (which act like placeholders) and fact symbols (which consist of everything in the fact type other than the terms).

Each fact symbol consists of one or more connectors, each being a contiguous set of words, either a verb phrase or a preposition.

Fact types can be classified by the number of terms involved, as:

- Associating a pair of terms using a verb phrase, to form binary fact type, for example:

  Employee **drives** Car

- Associating sets of three or more terms with a verb phrase and one or more prepositions, to form a ternary fact type, quaternary fact type, or other higher-order fact type, for example:

  Employee **passes** Project **to** Employee

- Associating a single term with an intransitive verb phrase, forming a unary fact type, for instance:

  Car **is hybrid**

Besides being categorized by the number of terms, fact types can be categorized by the relationship between those terms. This relationship gives us the following categories [3]:

- Associative fact types, which can be subcategorized as:
  a) Named relationship fact types (which can be either binary or higher order not unary). This type of fact type is used to link two terms with a verb phrase, for instance:

     “Car **has** Max Speed”

     The purpose of such fact type is to document the verb phrase to be used in any rule statements that need to refer to the relationship between instances of the two terms. Thus rule statements can include such sequences as:

"*Each* Employee ID *must* **identify** *one* Employee".

The same two terms may appear in more than one binary fact type, but only if the relationship between those terms is distinct from the last, for example, "Employee **reports to** Employee" and "Employee **trains** Employee".

Finally, Fact Models allow us to include the reverse mode of each fact type. If we pick the fact type "Car **has** ABS" the reverse mode of this fact type would be:

"ABS **is of** Car";

b) Partitive fact types (which can only be binary). A Partitive fact type is a particular variety of binary associative fact types in which the nature of the relationship is that each object signified by one term is part of an object signified by the other term. A verb phrase commonly used for Partitive fact types is "**is part of**", for instance:

"First Name **is part of** Name"

c) Attribute fact types (which can only be binary). This kind of fact type is useful to associate objects to their attributes. For example:

"Person **has** Address"

- Taxonomic fact types, which can be subcategorized as:
  a) Categorization fact types. Many of the terms we use are hyponyms, particular types of terms that can be associated to more general terms, or hypernyms. We may use the term 'Car ' and the term 'Truck' as well as the hypernym 'Vehicle'. 'Car' expresses all the characteristics and associations of 'Vehicle' but may have additional characteristics and associations expressed exclusively by it. This type of fact type allows us to represent taxonomic details about the concepts used in a business domain. For instance:

  "Car **is categorized as** Vehicle";

  b) Assortment fact types. Defines the set to which the individual concept signified by a proper name belongs. For example:

  "Subaru **is a** Car"

  By the fact that this kind of fact types associate proper names to terms, these are considered more as facts rather than fact types, implying a true sentence rather than a pro forma [3]. These kinds of fact types can be used for populating the business model;

- Finally, Fact Models allow the introduction of deriving fact types, fact types that are implied by other fact types, for example, if we add the fact types "Vehicle **has** Color" and "Car **is a category of** Vehicle" then we don't need to add the fact type "Car **has** Color" because the

model already knows that Car is a sub-category of Vehicle, and so it is implied that a Car has a Color.

2.4.1.2 Fact Model Dialects

It is well recognized that the quality of a database application depends critically on its design. To help ensure correctness, clarity, adaptability and productivity, information systems are best specified first at the conceptual level, using concepts and language that people can readily understand [32].

Fact based modeling has been researched and applied in business of semantic modeling for information systems since the 70s as a way to model, query and transform facts using attribute-free structures based on CNL. Subsequently, several developments have taken place in parallel, resulting in several fact based modeling "dialects", including NIAM, ORM2, CogNIAM, DOGMA and FCO_IM [33]. For this dissertation, and because of space limitations, this dissertation will focus mainly in ORM2 language.

These types of languages allow an organization to represent its business rules by different interconnected visual constructs. Such constructs might be facts and terms of the adopted fact model.

2.4.1.2.1 ORM 2 – Object Role Modeling

Object Role Modeling is a fact-oriented data modeling technique proposed by Terry Halpin in 1989 [34] for performing information analysis at the conceptual level, where the application is described in terms easily understood by non-technical users. In practice, ORM data models often capture more business rules, and are easier to validate and evolve than data models in other approaches [35]. Unlike the Entity-Relationship (ER) modeling and Unified Modeling Language (UML) class diagrams, fact-oriented modeling is attribute-free, treating all elementary facts as relationships.

This method simplifies the design process by using constrained natural language, as well as intuitive diagrams, which can be populated with examples, and by examining the information in terms of elementary facts, that is, atomic fact [32]. By expressing the model in terms of natural concepts, like objects and roles, it provides a conceptual approach to modeling.

ORM has been used productively in industry for three decades now. Recent case studies on the practical benefits of ORM in industry cover topics such as data quality firewalls [36], dynamic multidimensional denormalization [37], requirements engineering [38] and decision support systems [39].

2.4.1.2.1.1 ORM 2 Properties

Object Role Modeling views the world as a set of objects (entities and values) that play roles (their parts in each relationship). These are the most elementary elements of the ORM conceptual schema.

Entity type objects are "real world" objects that are identified by a definite description [32]. Each entity type has a reference scheme, that indicates how each instance of the entity type object may be mapped via predicates to a combination of one or more values [40].

Value type objects are used to represent constant values, so there is no requirement for a reference scheme in those cases.

To characterize the predicates that define the roles of each involved business term, ORM expresses these as named sequences of one or more role boxes, defining a separated box for each role. Both the normal fact type and its reverse form are included in the same declaration, separated by a slash the "/ " symbol.

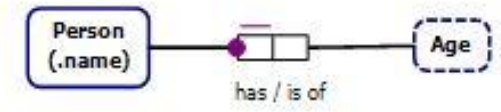


Fig. 2.4: Predicate expression with ORM

As it is possible to verify in the small ORM scheme, Person is an Object Type and is illustrated as a continuous line rectangle, having a reference scheme "(.name)". This means that a Person is identified by its name. The Term depicted as a broken rectangle is a Value Type and the middle rectangle is the predicate that connects the two terms. In this case, there are two rectangles, as there are two different roles represented (binary fact type), one for each direction of read. The defined fact types of this simple example are: "Person **has** Age" from left to right, and "Age **is of** Person" from right to left.

The bar over the first role is used to define an Internal Uniqueness Constraint. These constraints declare that instances for that role in the fact type population must be unique. With that said, it means that, for instance, in the last figure, the constraint on the "has" predicate verbalize that a Person cannot have more than one Age value but that it is possible for the same Age to be given to more than one Person.

Finally, to finish this first example, the dot placed on the "has" predicate is a Mandatory Role Constraint. What that constraint does is to define that each Person needs to have some Age value. It is a relationship that needs to exist in order for the business model to function properly.

These are the most basic functionalities of the ORM language, but ORM allows us to use many more constraint types [41]. Of course, the ORM language has a very vast list of possible constraints to be used, but irrelevant for this dissertation purpose, and so, the author will only be covering a small

portion of those constraints so that the reader can have a good perspective of the ORM language potential.

Besides the Internal Uniqueness Constraint, ORM permits some constraints like:

- External Uniqueness Constraints: Used to indicate that a combination of connected roles is unique;
- Subset Constraints: This constraint can be used to imply that a certain population that play a role must also play another role. For instance:

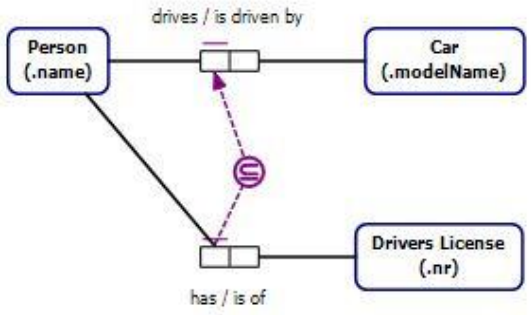


Fig. 2.5: Subset Constraint with ORM

This constraint indicates that a Person should only drive a Car only if "Person **has** Driver's License";

- Equality Constraint: Used to imply that the population between two different roles must be equal, and so this constraint can't be used if only one of the roles is mandatory;
- Exclusion Constraint: Useful to indicate that between all the involved roles, only one of those can happen at the same time;
- Inclusive Or Constraint: Implies that an object instance must at least participate in one of the associated roles [42];
- Frequency Constraint: Can be added to any role to specify the number of occurrences of this role by its object type.
- Subtyping Constraint: This kind of constraint permits the developer to define subtype entities. The subtype objects inherit the higher object characteristics and can have other exclusive characteristics:

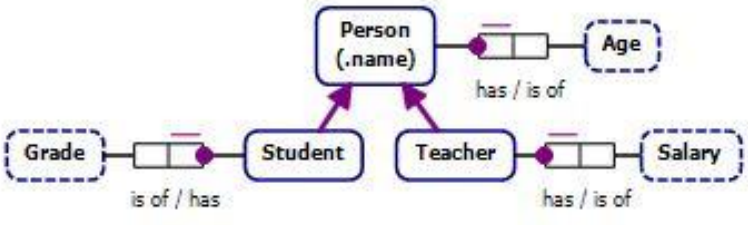


Fig. 2.6: Subtyping Constraint with ORM

ORM was designed to be semantically stable, expressive, and orthogonal [43]. The ORM modeling was developed around a set of principles. Each ORM model must facilitate the validation by domain experts, and so, this validation principle is achieved through 2 other principles, the population principle and the verbalization principle. The population principle determines that all structures must be easily populated with concrete examples. To satisfy this principle, an ORM model uses fact tables to display facts. The verbalization principle, in which all the elements of the model must be verbalized on a way that is intuitive enough for all the stakeholders, is satisfied by applying the principle of linguistic freedom, enabling facts to be expressed in a controlled natural language.

ORM uses a semantic stability principle, which defines that a fact should not suffer a modification as a consequence of other changes, unless the meaning of the same fact has been altered [43]. This is an advantage towards ER and UML model languages because, for instance, if we need to record a fact about an attribute, we would have to remodel this attribute. However, in a fact model, since it's attribute-free, this doesn't happen. Of course, this attribute-free architecture advantage comes with a price. The ORM attribute-free nature creates the tendency for the model to consume more space than the ER and UML models. This problem however can be resolved by using attribute-views on demand, like an ER automated model [41].

2.4.1.3 Fact Models Tools

The tools that have been research for modeling ORM are: VisioModeler, NORMA, ORM-Lite, DogmaModeler and Edraw Max.

2.4.1.3.1 VisioModeler

VisioModeler is an open-source, free platform for the representation of ORM conceptual schemas. It presents three distinct, yet integrated perspectives for the database design. The conceptual model is expressed in the ORM dialect and allows generating an automatically normalized logical model of tables and columns, selectable between IDEF1X or relational notation [44]. For the logical model the developer can generate a script that will either generate a new database as a DBMS catalog or modify an existing database.

Alternatively, VisioModeler can connect directly to the database server, through ODBC, and create or modify a database.

VisioModeler also has the capability of "re-engineering", extracting information from an existing database, and generate either the logical diagram (tables), or the ORM model.

2.4.1.3.2 NORMA – Natural ORM Architect

NORMA (Natural ORM Architect for Visual Studio) is a free and open-source plug-in for Microsoft's Visual Studio .NET developed mainly by Neumont University students, that supports the next generation of ORM (ORM 2). The tool supports entry of ORM2 schemas, verbalization of constraints, and code generation to a variety of DBMSs (including SQL Server, DB2, Oracle, PostgreSQL, and MySQL) as well as class models and XML schema [45].

NORMA is capable of importing ORM schemas entered in Visio for Enterprise Architects (VEA), but the diagrams need to be laid out manually.

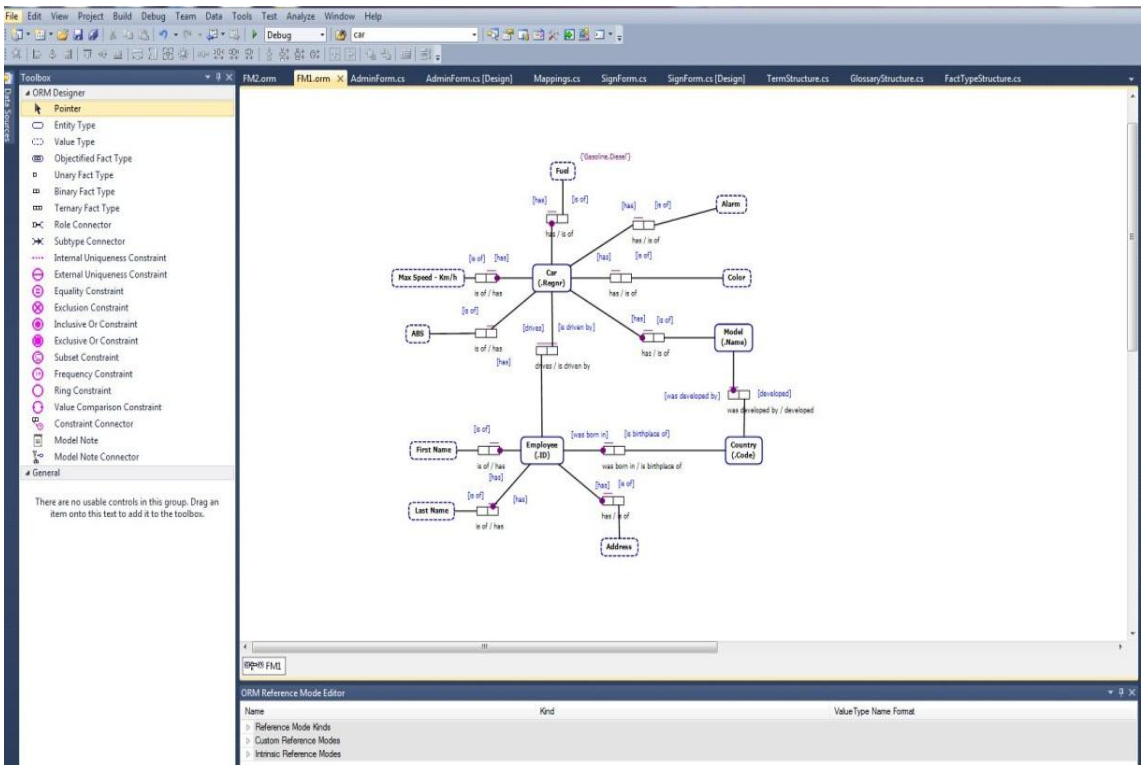


Fig. 2.7: Snapshot of the NORMA plugin on the VS Workspace

2.4.1.3.3 DogmaModeler

DogmaModeler is a java modeling tool, based on ORM. It supports ontology modularization and composition. This tool represents ontologies and can reason over them. DogmaModeler allows the user to create his own ORM Diagrams and to map from ORM to OWL language [46]. However, this tool was developed with the purpose of being used in a PhD and therefore not possible to run outside of the Vrije University Brussels in Belgium [47].

2.4.1.3.4 ORM LITE

ORM LITE is a light-weight, open-source modeling tool that supports ORM 2 notation. It was created as a self-learning environment to help popularize ORM. It can verbalize facts and generate relational models. It is written in Python and so it is multi-platform. It is cross-platform, working on Windows, Macintosh, Linux, and Unix [48].

ORM LITE includes the generation of SQL and Relational Maps. Indices are included for most uniqueness constraints. The application supports the entry of sample Data to populate each Fact Type and includes a Verbalizer window.

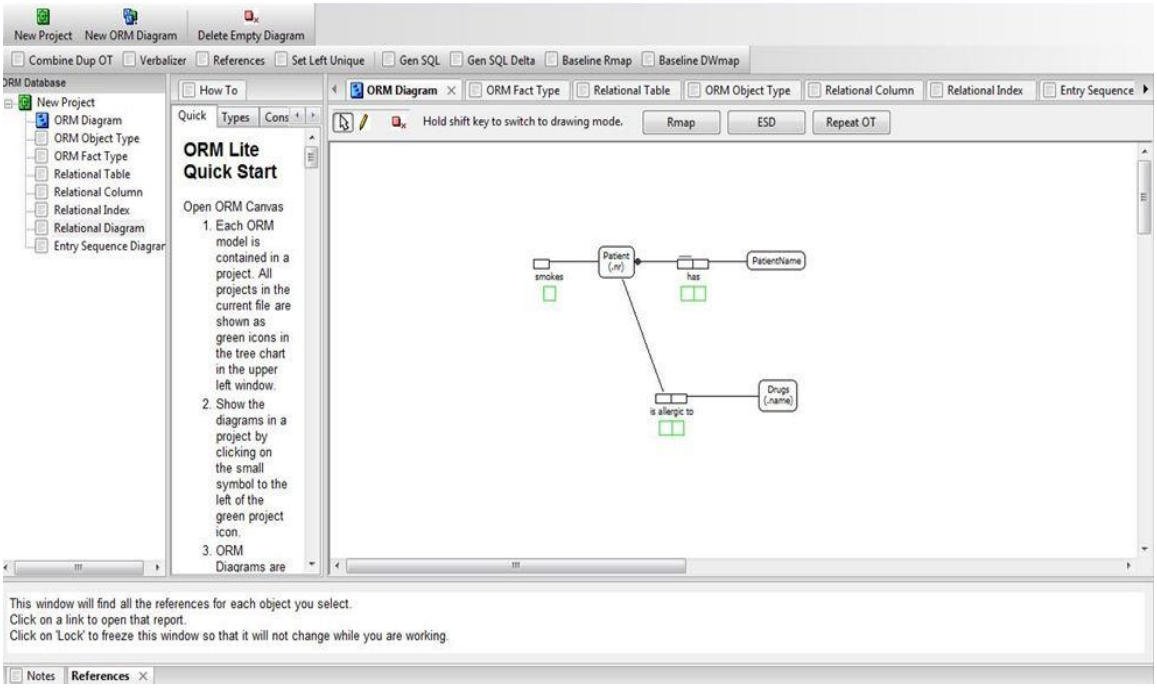


Fig. 2.8: Snapshot of the ORM-Lite Tool

2.4.1.3.5 Edraw Max

Edraw Max is modeling tool developed by Edraw Soft that allows the construction of various types of diagrams, including ORM diagrams for database design [49]. This tool is capable of doing binary, ternary and quaternary fact types and many types of constraints, like the ring constraints and frequency constraints, however this tool doesn't have a verbalization tool and neither has an API to allow the user to programmatically manipulate the designed ORM diagram.

The full version of the tool is paid, however it is possible to download a trial version, free of charge from the Edraw Soft's website.

2.4.2 Ontologies

Ontologies are knowledge models used to represent a set of concepts within a domain and the relationships among these concepts, similarly to the Fact Model. However, the ontologies represent these concepts in a rather distinct way. In [50] , an ontology is said to be a shared, formal and explicit understanding of some domain of interest, which may be used as an unifying framework to solve interoperability issues between enterprises.

Instead of using fact types and terms to express the business domain of discourse, the ontologies rely on a set of classes, properties and individuals, promoting the business conceptualization.

Ontologies have been used largely for data base design, and software applications that need shared information, where their information domains are related to a particular area of knowledge. One very practical example is the growing use of ontologies in semantic web. The ontologies are used in this area as an infrastructure enabler, working as specifications of the conceptualizations at a semantic level [51].

2.4.2.1 OWL – Web Ontology Language

The most popular used ontology language is the Web Ontology Language, or OWL. OWL was defined as the standard modeling language for semantics in the World Wide Web Consortium (W3C) [52]. The OWL ontology is a Resource Description Framework (RDF) graph constructed with a set of triples. These triples consist of three parts, a subject, a property and an object. Fig. 2.10 illustrates an example of a triple:

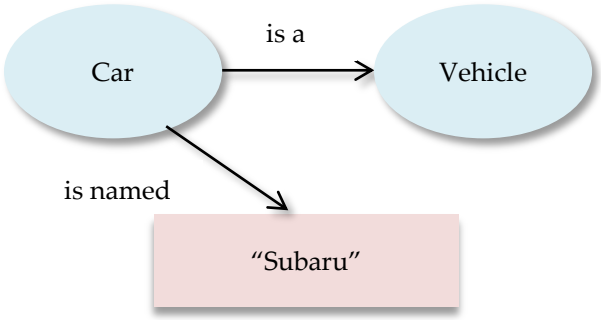


Fig. 2.9: Ontology Triple Example

OWL is able to perform reasoning on triples because it is based on Description Logic. DL is a fragment of the first-order predicate logic and the formalism for representing knowledge [47].

As it has been already mentioned in the last subsection, the ontologies rely on a set of elements, denoted as classes, properties and individuals. A class represents a set of several resources that share common characteristics or are similar in other ways. The most general class in OWL is represented has "owl : Thing". All the individuals created in the OWL environment are part member of this general class. The properties are used to describe a resource, establishing the relationships between subjects and objects. Finally the individuals are the resources that represent an instance of the class.

2.4.2.2 Main Differences between FM & Ontologies

Both the ORM and the OWL language are used for modeling information. Although the fact that both of these languages have the same common purpose, they have their differences. ORM uses an Unique Name Assumption, that is, it assumes that two concepts with different names are different and two concepts with the same name are equal. In other hand, the OWL language can't make that kind of assumption unless it is explicitly stated [53]. This characteristic of the OWL language makes it easier for information systems to handle the data ambiguity.

Another difference between the two languages is the reasoning feature in the available tools for each of the languages. The available OWL language reasoners can perform consistency checks on the semantic of the model in addition to adding information that is only implicitly stated. On the other hand, there is no ORM tool that does this. There are ORM tools available that perform consistency checks on the syntax, but none that perform consistency checks on the semantic.

Finally, the OWL language has an Open World Assumption, which means that in OWL, if knowledge is not found, it doesn't mean that it is false, but rather unknown to date. This feature is very useful in a semantic web strand because the knowledge is constructed by each individual, so it is assumed that knowledge isn't all available at once. ORM language on the other hand, has a Closed

World Assumption, that is, it assumes that the information that is out of its knowledge base is false. This feature is very useful when designing a database.

Both languages are structurally very similar to each other. In [47], it has been concluded that almost all the concepts and constraints in ORM can be represented in OWL, proving that it is indeed, very possible to map one language to another, although the non-existence of tools for that same purpose.

ORM is the most suitable language for modeling an information model, since the restrictions on the OWL classes don't need to be met (Open World Assumption), which makes the information models created in OWL very error prone.

2.4.3 The Decision Model

The Decision Model is an intellectual template for perceiving, organizing, and managing the business logic behind a business decision, where business logic is the means by which the business derives conclusions from facts [54]. As an intellectual template, the Decision Model is a logical representation of business logic. It is, by deliberate intent, not a physical model of how that business logic relates to a specific technology. Instead, it is an intellectual template for the full and rigorous specification of that logic [55].

The Decision Model is not simply a list of business rules or business statements. Rather, it is a model representing a structural design of the logic embodied by those statements [55]. The Decision Model can be anchored to any and all other kinds of models (data models, fact models, process models), but maintained independently of them.

Indeed, the main purpose of creating a Decision Model was to be able to make the business logic independent of the rest of the business architecture, such as, data, process, or the used technology, enabling a more reachable way of storing and modifying the involved business rules.

2.4.3.1 Building a Simple Decision Model

The simplest element of a Decision Model is a two-dimensional table, called Rule Family. The Rule Family is used to relate a set of conditions to one, and only one conclusion. To better explain how a Decision Model functions, a simple example will take place using the open-source tool OpenRules [56].

A Rule Family is simply a set of atomic business statements, grouped by their conclusion fact type.

RuleFamily VehicleType							
Condition Fact Types						Conclusion Fact Type	
Number of Axes		**Vehicle Class**		**Number of Wheels**		**Vehicle Type**	
Is	2	Is	C1	Is	2	Is	Motorcycle
Is	3	Is	C2	Is	6	Is	Bus
Is	2	Is	C1	Is	4	Is	Car

RuleFamily VehicleClass					
Condition fact Types				Conclusion Fact Type	
Number of Axes		**Vertical Height of the 1st Axe**		**Vehicle Class**	
Is	2	Is	< 1,1m	Is	C1
Is	3	Is	> 1,1m	Is	C2

Table 2.2: RuleFamily for Vehicle Type Classification

This small example represents the classification of a vehicle by its type. It is possible to observe that the first Rule Family has three Condition Fact Types, all ANDed, and that the condition "Vehicle Class" depends on the decision of another Rule Family, denoted as "RuleFamily VehicleClass". To execute a Rule Family, the OpenRules Tool uses the command [56]:

Define VehicleType := VehicleType ()

Like the Fact Models, the Decision Model has the advantage of representing each row into a sentence that is written in natural language, making it easy for business audience to comprehend.

Besides the Rule Families, the Decision Model has a higher level element, named Decision Model Diagram. These Diagrams are used to illustrate the Decision Model's structure and the detailed content of its Rule Families [55]. Following, a diagram for the last Rule Families is presented:

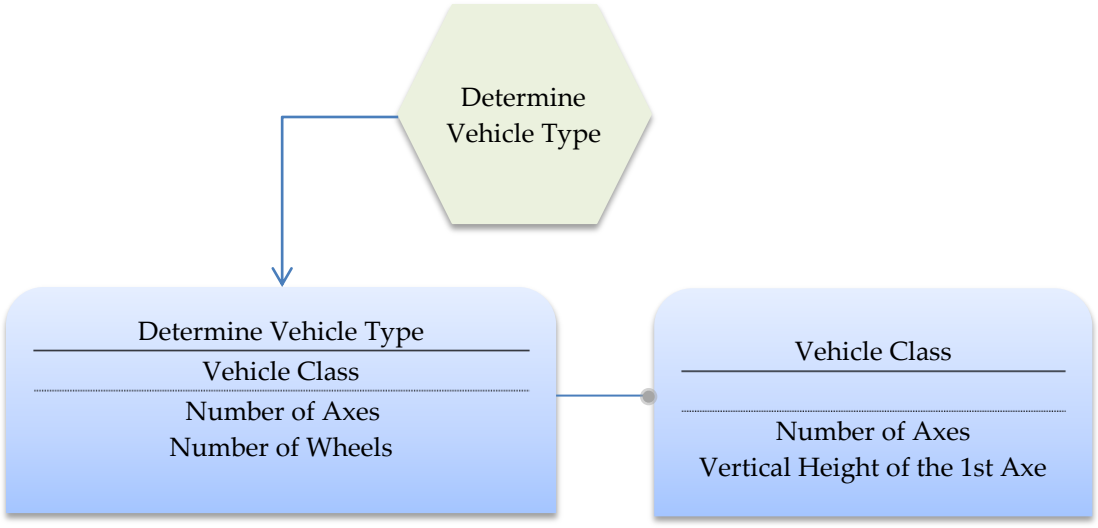


Fig. 2.10: Decision Model Diagram

The Decision Model diagram begins with an octagonal shape that represents the entire business decision. The other shapes in the Decision Model diagram represent Rule Families. The Rule Family that is directly connected to the business decision shape is called the Decision Rule Family, since its conclusion answers the question made by the determiner.

Each of the Rule Families is represented by a solid line and a dashed line. All the labels between these two lines represent Conclusion Fact Types from other Rule Families. On the other hand, all the labels below the dashed line represent the Condition Fact Types from that Rule Family, not being related with other Rule Families.

2.4.3.2 Relationship between Decision Models and Semantic Models

The Decision Model offers structural and semantic principles to establish the integrity of a business decision and allows retaining its coherence regardless of the size of the model.

Semantic Models, such as the Fact Models or the Data Models, can be viewed as representations of the information presented in a Decision Model, as conditions and conclusions. These Fact Types should be connected to a Fact Model, a Data Model, or at least a glossary of Fact Types. The connection of a Fact Model to a Decision Model can be especially interesting, since the Fact Model offers a conceptual view, allowing the users to familiarize themselves with the business terminology and the Model general behavior. That said, the Fact Model can be used to populate the Rule Family Tables of the Decision Model

Because the Decision Model is based on the inherent nature of business logic and because business logic is closely bound to the other models in requirements, it is natural that all models connect together, sharing metadata [55].

2.5 Concluding Last Remarks

In this chapter an overview of the most important concepts behind knowledge representation of business domains was presented. With this study, the author now has the necessary tools in order to be able to develop a viable solution to prove the presented hypothesis. The Fact Models were chosen has the preferable business models to solve the given problem, as it provides the necessary formalization and conceptualization of the business domain and at the same time, providing an easy to understand vocabulary so that the business stakeholders can be able to cooperate on the design and creation of the domain of discourse. On the other hand, the ontologies can be used as good business domain creation tools, but the fact that this models rely on attributes, makes the Fact Models a better solution for the automation path.

3

Establishing an Enterprise Reference Lexicon

This chapter as the following structure: First, a brief introduction to the problematic of systems interoperability is presented, introducing the main concepts necessary for the semantic interoperability resolution and exploring the maturity levels for interoperable business systems. Next, the MENTOR methodology is presented, demonstrating its main capabilities. Finally, the author presents the proposed architecture adaptation of this methodology along with a description of all the particularities behind that adaptation.

3.1 Interoperability between Business Information Systems

Over the years, it has been confirmed an increasing demand for information models to interoperate by exchanging data. For these data exchanges to be meaningful it is essential that the business information requirements that are met by the data stored in these systems are understood so that suitable data exchange mechanisms can be developed [57].

But what is interoperability exactly? IEEE [58] has defined interoperability as the ability of two or more systems or components to exchange information and to use the information that has been exchanged. In [59], interoperability is described as the ability of a set of communicating entities to exchange specified state data and operate on that state data according to specified, agreed-upon, operational semantics. Hence, interoperability can be depicted simply has the ability of two or more systems to stay connected to each other, being able to exchange information even though they might be in heterogeneous environments and operating on specified, standard operational semantics.

For a set of information systems to be able to reach proper interoperability, it is necessary that several interoperability degrees are taken into consideration. Obrst, in [60], indicates that, because data

is typically considered in terms of its representation and usually its declarative representation, it can be considered as having the properties of that representation, and so, it can be considered that the data needs to be syntactically, structurally and semantically interoperable.

At a structural level, the encountered mismatches will be related to the chosen arrangement of the concepts and their relationships as for the syntactic level, this heterogeneity will depend on the kind of formalism used, that is, the implementation languages, interfaces, and execution platforms used by each system [61].

As it has already been stated in chapter 2.1, the existence of semantic mismatches is quite common, as it is very usual to encounter similar words or sentences with different meanings, creating unwanted ambiguities. With that in mind, at a semantic level, data structures capable of capturing the agreed semantics of the domain are needed. As examples of that kind of structure we have, the already mentioned Fact Models and Ontologies (OWL), but also Entity-Relationship Model (ER), Unified Modeling Language (UML), Object Constraint Language (OCL), etc.

In order to achieve a proper interoperability between distinct systems, it is said that information must be physically exchanged (technical interoperability), must be understood (conceptual interoperability) and must be used for the purpose for which it has been produced (conceptual and organisational interoperability) [62].

Similarly, the research conducted by the Virginia Modeling Analysis & Simulation Center derived an interoperability layered model, the Level of Conceptual Interoperability Model (LCIM) [63]. The LCIM defines the level of maturity of a system in terms of interoperability:

Level 6 **Conceptual Interoperability**	**Composability**
Level 5 **Dynamic Interoperability**	
Level 4 **Pragmatic Interoperability**	**Interoperability**
Level3 **Semantic Interoperability**	
Level 2 **Syntactic Interoperability**	
Level 1 **Technical Interoperability**	**Integratability**
Level 0 **No Interoperability**	

Table 3.1: Levels of Conceptual Interoperability Model

The different levels are characterized as follows:

- Level 0: This level defines the state of maturity where a system doesn't present any kind of interoperability what so ever;
- Level 1: Systems at this level of maturity are systems that share a common communication protocol for exchanging data between them. On this level, a communication infrastructure is established and the underlying networks and protocols are unambiguously defined;
- Level 2: On this level, a common protocol to structure data is used defining an unambiguous structure data formalism;
- Level 3: If two or more involved enterprise systems use a common interface exchange model, then the level of Semantic Interoperability is reached. On this level, the meaning of data is shared;
- Level 4: This level of maturity is reached when the interoperating systems are aware of the methods and procedures that each system is employing, that is, the purpose and context in which the information is exchanged is understood by all the involved enterprise systems;
- Level 5: If the involved systems reached this level of maturity, that means that one system can react to changes made by another participating system;
- Level 6: Finally, systems that reached this level are systems that rely and document conceptual models based on engineering methods enabling their interpretation and evaluation by other engineers. Models like the Fact Model and Ontologies fulfil this requirement.

Many methodologies focused in enterprise interoperability have been purposed over the years. These methodologies can be either, the implementation of a common standard (ISO, STEP), which will force all the involved stakeholders to migrate to that same standard, or the use of a methodology, such as MENTOR (Methodology for Enterprise Reference Ontology Development), that operates as an intermediate between each of the source data models, allowing the involved stakeholders to maintain their own business models.

Indeed, forcing all the stakeholders to adopt the same common Business Model, does not work in most of the cases. Thus, an ideal solution would be to keep the terminologies and the classifications used by each stakeholder and use a reference structure as the mediator in the communications between them. Additionally, the introduction of new reference business structures would enrich the community and each enterprise should feel more motivated to be part of the group, with the possibility to keep their own definitions [64].

3.2 MENTOR Methodology

MENTOR [64], is a methodology that allows the construction of an ontology of reference in a collaborative environment, making it possible for every involved stakeholder to share their knowledge and combine it in order to come to an agreement about all the terms and relations that will be included in the business domain. Besides allowing a cooperative ontology building, this methodology allows the construction of ontologies by scratch, ontology reengineering and ontology merging.

The MENTOR methodology is composed by two phases that have three steps each. A diagram of this methodology is presented in Fig. 3.2 [64]. The first phase of this methodology, named Lexicon Settlement Phase, consists on the knowledge acquisition phase. This phase is comprised in the following steps [64]:

- **Terminology Gathering:** In this step, MENTOR receives all the relevant and necessary terms from all the involved stakeholders. MENTOR tags each of this terms with the name of the respective contributor, so that it is possible to know in the further steps which of the stakeholders contributed with what terms;
- **Glossary Building:** With all the terms gathered, a description of each of these terms is requested. Each of the participants manually provides a description for each of the gathered terms. When the participants come to a suitable description for each of the terms a voting process takes place. In this process, all the participants must review the descriptions uploaded by all of the other participants and vote for the most suitable one. After the voting process, if all the stakeholders came to an agreement, the terms are established as the reference terms and a glossary is built (**O2**). On the other hand, if the stakeholders didn't come to an agreement, the semantic mismatches (**O1**) are recorded for future ontology mappings an the methodology goes back to the term revision sub-step;
- **Thesaurus Building:** In the final step of the first phase, a taxonomic structure is defined. Every single term in the glossary is classified in a way to properly place it in the taxonomic tree. Once again, if there is an agreement between all the stakeholders, a thesaurus is defined (**O3**) and the methodology advances to the second phase, if not, the methodology goes back to the beginning of this step.

The second phase or Reference Ontology Building Phase is where the reference ontology will be built. Similarly to the first phase, three steps constitute the second phase [64]:

- **Ontologies Gathering:** In this step, ontologies or other types of knowledge representation models are gathered;
- **Ontology Harmonization:** The structure of the reference ontology is discussed in this step, taking into account the structure of the previously defined thesaurus. If an agreement is reached, then the taxonomy structure of the reference ontology is established (**O4**) and the

second cycle of this step begins. In this second cycle, the contents of the gathered ontologies are harmonized using the semantic mismatches from the previous terms revision. When the stakeholders agree with each other, the reference ontology is established (**O5**) and the methodology can advance to the final step;

- **Ontologies Mapping:** In this final step, mapping tables (**O6**) are created, describing the relationships between the reference ontology and the ontologies from all the involved enterprises;

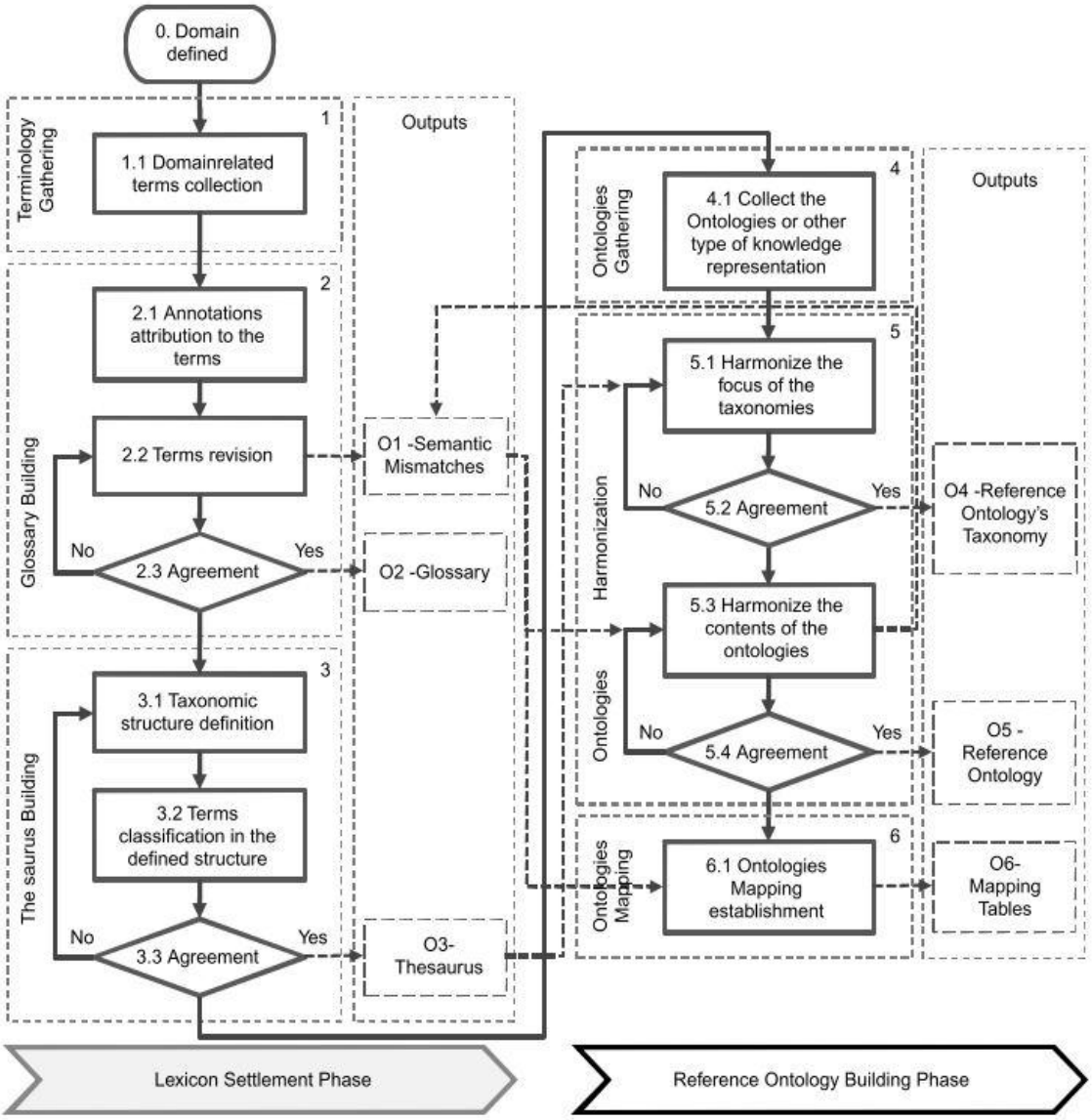


Fig. 3.1: MENTOR Methodology [64]

The last version of this methodology was conducted by Gaspar in [51], where he successfully enriched the MENTOR methodology using qualitative information collective methods. The developed

prototype provided an implementation of the first term (Terminology Gathering), passing all the gathered terms to a Protégé plugin that provides collaboration methods suitable for the next steps.

3.2.1 Mediator Ontology

Since there will be, much likely, a large set of semantic mismatches during the MENTOR methodology, a Mediator Ontology (MO) was developed to aid the Ontology Mapping Step of MENTOR, serving as a reference for mediating the mapping establishment and its subsequent reasoning [51].

Besides the previous capabilities, the Mediator Ontology is able to represent ontology semantic operations such as, the semantic mismatches found in the Glossary Building step, the semantic transformations identified in the Harmonization process and other ontology operations [51].

To represent the ontology operations, Agostinho proposed a five-tuple mapping expression [65] :

$$\boldsymbol{MappingTuple(MapT): < ID, MElems, KMType, MatchClass, Exp >}$$

Each of the 5-tuple elements has the following purposes:

- ID is the unique identifier of the Mapping Tuple;
- MElems is the pair (a, b) that indicates the elements that are being mapped;
- KMType stands for Knowledge Mapping Type. The mapping relations are related to a traceability element, that is, it relates one of the defined terms on the reference vocabulary to one term on one of the participating business vocabularies:

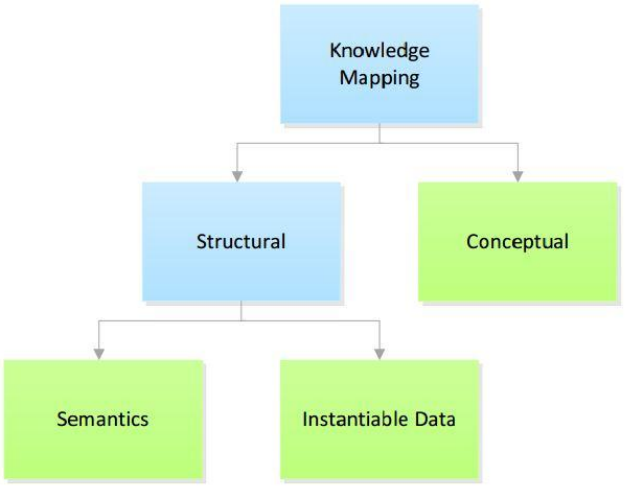


Fig. 3.2: KMType Value

The KMType can be classified as “Conceptual” if mapping concepts and terms; as “Semantics” if mapping model schemas; or as “Instantiable Data” if mapping Instantiation Rules;

- MatchClass stands for Match/Mismatch Classification. The MatchClass value will depend on the KMType value:

KMType	MatchClass
Conceptual	- LessGeneral - MoreGeneral - Language - Equal - Disjoint - Naming
Semantics	- Granularity - SubClassAttribute - SchemaInstance - Encoding - Content - Precision - Abstraction - Structuring - Equal - Disjoint - Naming - Coverage
InstantiableData	- PhysicalDependency - Uses

Table 3.2: MatchClass Values

- Exp stands for the mapping expression that translates and specifies the previous tuple components

In order to make all these mapping information processable by computers, a dedicated Knowledge Base needs to be developed. For that matter, the Mediator Ontology was developed as a Java application and prepared to store mappings from OWL files. The Fig.3.5 shows an UML schema that represents the most recent Mediator Ontology structure.

For this dissertation, the author will adapt this same structure and prepare it for the storage of ORM file mappings.

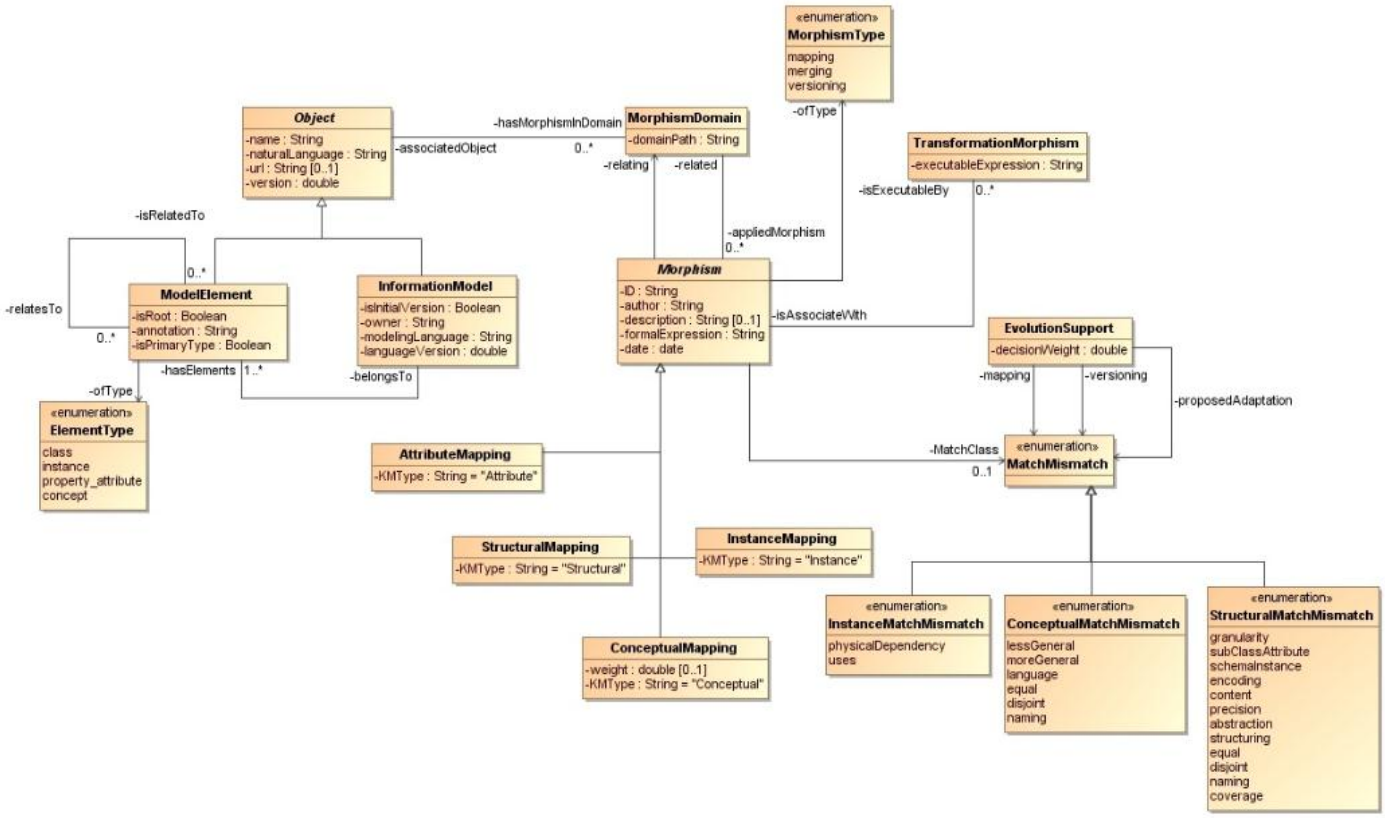


Fig. 3.3: UML MO Structure [65]

The Mediator Ontology structure has two main classes, named “Object” and “Morphism”. The class “Object” represents any “InformationModel”, which is the model itself, and “ModelElements“ (also belonging to the IM) that can either be any of the model´s elements. The class “Morphism” associates a pair of “Objects” (related and relating), and classifies their relationship with a “MorphismType”, “KMType” (if the morphism is of the type mapping), and “Match/Mismatch” class.

With the mappings stored in the KB (Mediator MO), all the information regarding the mappings between the models of business partners can be accessed by their local systems. The mediator takes responsibility on the translation from one message format to another allowing the communication between two different systems. To put into perspective, Fig 3.6 illustrates the communication between two enterprises with the mediator in the middle.

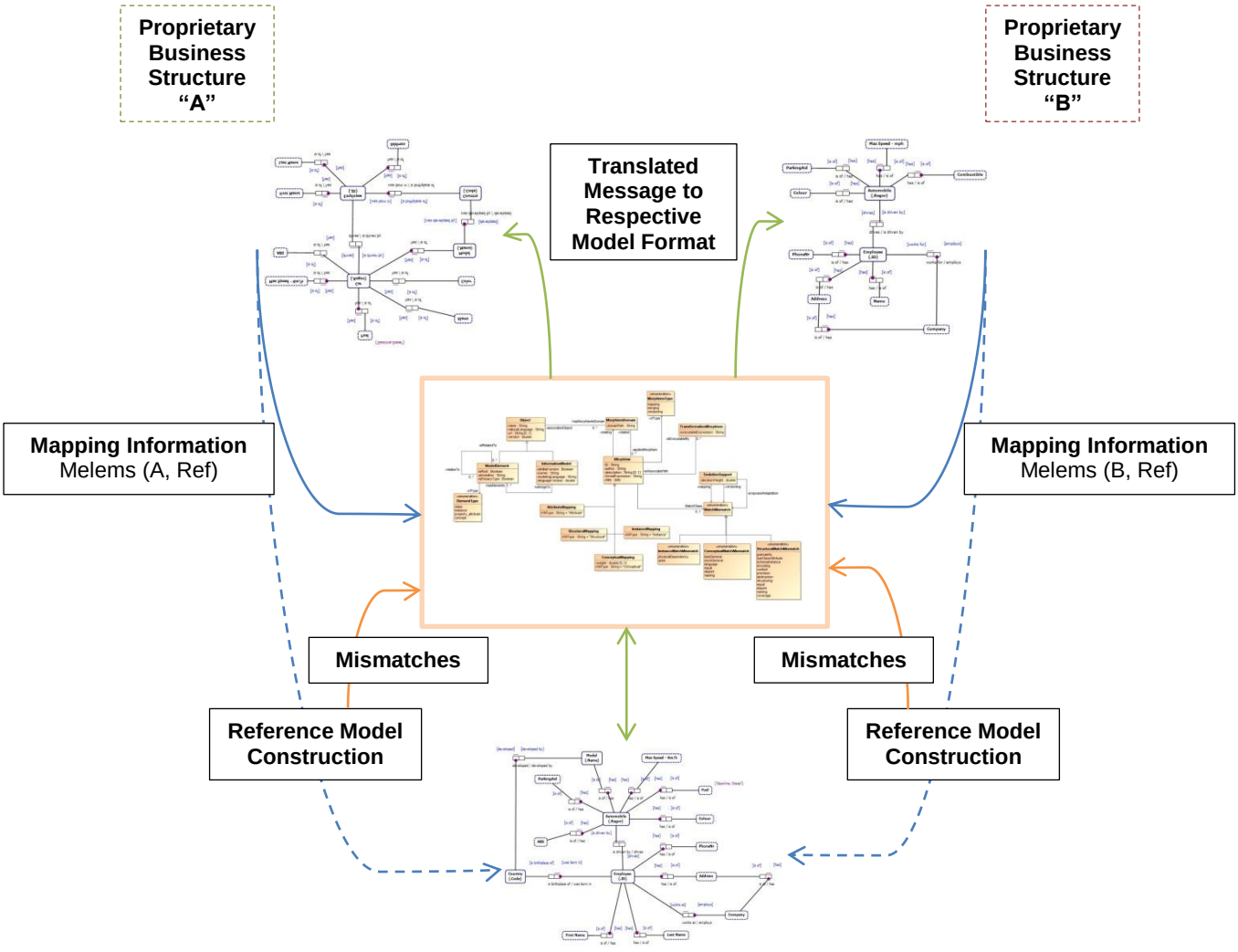


Fig. 3.4: Data Exchange with the Mediator Ontology Aid [51]

When the MENTOR methodology is first used, there are no mappings between each of the enterprise models. The involved stakeholders need to build a common reference business vocabulary, using the mediator ontology to establish mappings between the reference model elements and each of their own model elements.

After establishing all the mapping relations, when one of the business partners tries to communicate with another partner, a message needs to be sent to the mediator. The mediator is then responsible to transform that message in the destination partner format and to deliver it.

3.3 Concluding Last Remarks

In this chapter the problematic provoked by the necessary business interoperability was presented, allowing to understand the various maturity levels that an interoperable system needs in order to function properly with others. As a solution to this problematic, many systems have been developed, including the already mentioned MENTOR. The author finds this a proper methodology to solve the interoperability problems and so, the author will present in the next chapter the developed solution based on this methodology.

4

Proof-of-Concept Implementation

Now that enough knowledge about Fact Modeling and Data Extraction Methods was gathered, the author is in position to propose a solution for the adaptation of the MENTOR methodology in order to be able to establish a reference lexicon and semantic mapping representations between business models, providing a first contribution towards the automatism of the methodology.

This chapter is structured as follows: In the first section, the technologies that were chosen for the implementation of semantic mapping representations and reference lexicon settlement are presented. Furthermore, the author exposes the architecture and description of the solution for the given problem. Finally, a detailed description of the developed methodology is shown.

4.1 Used Technology

This subsection is used to present all the necessary tools used for the development of the reference lexicon building tool, showing their respective descriptions and manifesting their utility for the given problem.

4.1.1 .NET Framework

The .NET Framework is a software framework developed by Microsoft. The .NET Framework, similarly to the Java platform has the advantage of allowing one to write code for the .NET platform instead of being restricted to a specific system or device. This property of the .NET framework is possible due to a .NET Framework software, known as Common Language Runtime (CLR), an application virtual machine used to execute the written code. The CLR serves as the execution engine of the .NET Framework, providing services, such as, memory management, security, and exception

handling [66] and capable of executing 33 different languages, such as, C# and J# [67], a java-based language.

The .NET Framework uses a large class library, known as FCL, or Framework Class Library, and provides language interoperability, that is, the used language can use code written in alternative languages.

The .NET Framework was chosen as the ideal work environment for this project due to the fact that the NORMA tool is a plug-in for the Microsoft Visual Studio that provides an API that allows the developer to edit and manage programmatically ORM Fact Models.

4.1.2 NORMA

The NORMA tool was already presented in the subsection 2.4.1.3.2 of this document, and so there is no relevance to introduce it once more. However, it is important to explain the reasons why this tool was chosen, instead of the other competitors. The NORMA tool was chosen mainly because, it provides the most complete support for the ORM 2 notation, supporting unary, binary, ternary fact types, subtyping constraints and even objectifications. In addition, the NORMA tool provides an open-source ORM API that allows one to programmatically manipulate and manage an ORM model inside a C# project.

Besides this capability, NORMA is able to generate and map the ORM model to a variety of implementation targets, such as database engines, object-oriented languages and XML schemas, proving to be a CASE tool for the ORM language.

4.1.3 NHunspell

NHunpell is a free Open Office spell checking, hyphenation, word stemming and thesaurus library based on Hunspell. NHunpell has an available API for the .NET Framework, providing C# libraries that allow the developer to check vast dictionary.

This tool was chosen for this project, in order to provide a more accurate way of detecting similarities between the terms of each loaded models in the methodology.

4.1.4 MySQL

MySQL is a popular open-source relational database management system (RDMS). It is widely used in web applications, and is a central component of the widely used MAMP (Microsoft, Apache, MySQL, Perl/PHP/Python) open source web application [68].

MySQL uses the SQL language (Structured Query Language) to perform operations on relational databases. With this technology, the author is able to build databases in order to store the output reference model data and semantic mismatches before passing them to the mediator ontology.

The MySQL databases can be connected to the C# language by using a set of Visual Studio's features named LINQ (Language-Integrated Query). These features extend query capabilities to the language syntax C# [69].

4.1.5 Protégé

Protégé is a free, open source ontology editor and management tool that allows the construction of knowledge-based applications with ontologies. The Protégé tool allows to export and import ontologies in the language OWL (OWL 2 since the version 4.0) and XML/RDF, Turtle and Manchester syntax. The output of this tool is not proprietary and can be viewed in standard editors.

The Protégé tool also provides a java-based API that allows a user to programmatically manage the ontologies. This tool was chosen by the author, in order to be able to transfer the semantic mismatch information and to be able to communicate with the reference model.

4.1.6 RESTful Service

REST (Representational State Transfer) is the underlying architectural principle of the web. It consists in a set of architectural constraints applied to components, connectors and data elements and ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements.

REST can be built in any programming language that can handle HTTP requests. It is an architectural style that can be used to build software in which clients (user agents) can make requests of services (endpoints). REST is one way to implement a client-server architectural style [70].

The REST constraints are based on the same principles that govern the Web [70]:

- Each client interacts with resources, which are anything that can be named and represented, being addressed via a URI (Unified Resource Identifier);
- The interaction with resources, located by their unique URIs, is made by using the standard HTTP verbs, such as GET, POST, PUT, etc.;
- The declaration of the resource's media type is used with the HTTP Content-Type header (XML, JSON, JPG, etc.);

- All the information necessary to process a request on a resource is contained is contained inside the request itself;

The reason why this service is needed for the development of this project is that, it enables the connection between the reference lexicon building methodology (developed in the .NET Framework using the C# language) and the Mediator Ontology (that was developed in the Java language). This way, we are able to transfer resources, such as the semantic mismatch information to the Mediator Ontology.

4.2 Lexicon Settlement

For the construction of a methodological approach for building a domain reference lexicon, some adaptations to the first 3 steps of the MENTOR methodology were carried out.

The Terminology Gathering Step (Step 1) is the step where all the business models knowledge is gathered, and so the author decided to use Fact Models for that matter, since these models allow a formal and conceptual representation of the business knowledge.

For the Glossary Building Step (Step 2), knowledge extraction methods will be used, using a proper tool, like NORMA and the NORMA API to programmatically extract and manage the fact model elements.

Finally for the Thesaurus Building Step (Step 3), the author will develop a program that manages the extracted information on the last steps and defines the taxonomic relationships between each of the extracted terms, automatically defining the thesaurus.

4.2.1 Defining the Domain

One of the main purposes of this thesis is to prove that the use of a business knowledge based on fact models increases the computational intelligence of its information systems. With that said, the first step in this methology adaptation will be to define the domain and the extraction of its concepts.

For that matter, Visual Studio's NORMA was chosen as the ideal tool, as it allows the construction of a formal and conceptual model and the posterior data extraction by programmatically manipulating each model elements with the aid of the NORMA's API.

4.2.2 Glossary Building

After gathering all the involved terms and relationships from the Fact Models that were introduced in the previous step of this methodology, it is time to establish the glossary.

Knowing that the glossary consists basically on an ordered list of terms, with the respective descriptions, we can use the conceptual characteristics of the fact models, filling the glossary file with each of the involved terms. To extract the definitions of each term, we can rely on the fact types and constraints present in the fact model or proprietary note descriptions given by each enterprise.

NORMA makes this extraction possible because it contains a tool, named as "ORM Verbalization Browser" which takes the conceptual information of the model and creates a formal query with that information. The following figure, illustrates well the capability of this tool in NORMA, showing all the information known about the "Automobile" Term, including every single relationship of it with other terms.

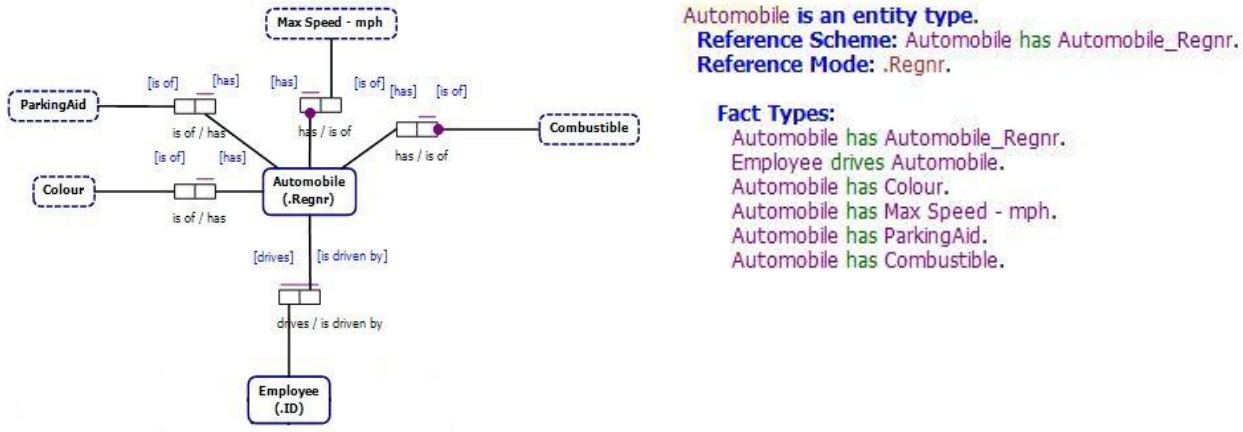


Fig. 4.1: On the Left: ORM Model Schema. On the Right: Verbalization Browser Tool

After extracting the concepts from each of the Fact Models, a 2-phase process for semantic mismatches detection takes place. This process is divided in two phases: First, the gathered elements pass by a Term/Definitions revision. This process can have four kinds of mismatches [71]:

- **Case 1:** Existence of 2 syntactically different terms with the same meaning description. In this case we will have to choose one of the terms for being the reference in such semantics meaning. These kind of mismatches need to be recorded for future mappings;
- **Case 2:** Existence of 2 syntactically equal terms with the same meaning description. Here we erase one of the 2 terms;
- **Case 3:** Existence of syntactically different terms with 2 different meaning descriptions. This case is to be ignored, the 2 terms are maintained;

- **Case 4:** Existence of 2 syntactically equal terms with 2 different meanings descriptions. In this case, we consolidate all the provided descriptions together in one of them and erase the other. It should be possible to change the name of this term, changing it to a preferable one. Once again, these kind of mismatches need to be recorded.

The second phase of semantic mismatch detection passes by the detection of synonyms, using the myThes tool provided by the NHunspell API.

During the process of choosing the final terms and definitions that will take part in the domain reference lexicon, a list of the semantic mismatches records is established, providing a link between the proprietary terms to the reference terms. These semantic mismatches are passed and stored in the Mediator Ontology.

Finally, after having all the reference terms and definitions chosen, the Glossary is built and stored in a Data Base using MySQL.

4.2.3 Thesaurus Construction

After having the glossary built, we can step forward to the thesaurus definition. In this process, the terms present in the glossary will go from a list to a hierarchical structure, in other words, taxonomy.

Since we are talking about hierarchical structures, we must take into account that taxonomies follow the same type of relation to each of its involved terms. That said, in order for the user to be able to obtain a Thesaurus, the user must first indicate the type of Thesaurus that is intended. The type of the Thesaurus is basically related to the role of a fact type. For instance, if we were to use a fact model that included the fact type "Car **has** Color", then the Thesaurus generated using the type "**has**" would have a father member "Car" and a son member "Color". On the other hand, using the fact type "Car **is driven by** Employee" and generating the same kind of Thesaurus, we would obtain no member.

So the Thesaurus is generated by analysing the fact type relationships between each of the reference terms, assuming that every mismatch was taken care on the previous step.

4.3 Architecture

In order to develop a methodology for building a reference lexicon, the author came up with a concise architecture, using the already mentioned tools and services. The following figure illustrates a general overview of the tools architecture and relationships:

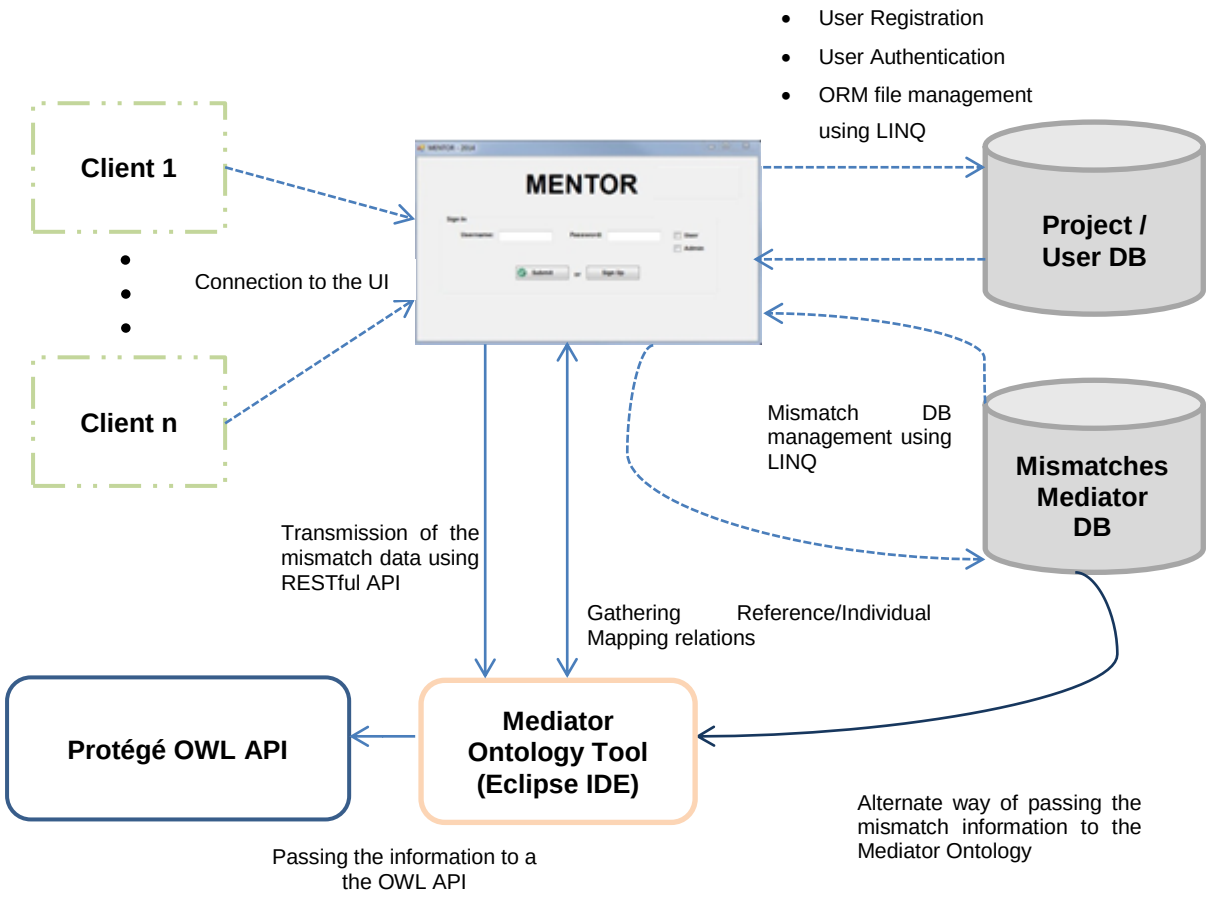


Fig. 4.2: Reference Enterprise Lexicon tool architecture

This architecture is composed by 5 main components:

- The MENTOR User Interface (Visual Studio environment) where the Fact Models are gathered and manage;
- The Project/User Data Base used to manage the user information and the projects that these users took part;
- The Mismatch Mediator DB used to store the mismatches obtained on the Glossary Building Step;
- The mediator ontology (Java Eclipse environment), used for storing the mismatches and for subsequent consultation of the mapping relations between the reference to the proprietary terms;
- Protégé OWL API used for the representation of the business model in a ontology conceptualization.

4.3.1 Project/User Data Base

The Project/User Data Base was develop for storing: user identification, serving as a means of user authentication on the user interface; projects created by each user; fact models (ORM files), which are connected to a certain project and a certain user. This way, the user interface will be able to identify the different fact models by its contributor ID. Figure 4.3 shows an example of the created DB using the MySQL tool, featuring in this case the ORM Files table:

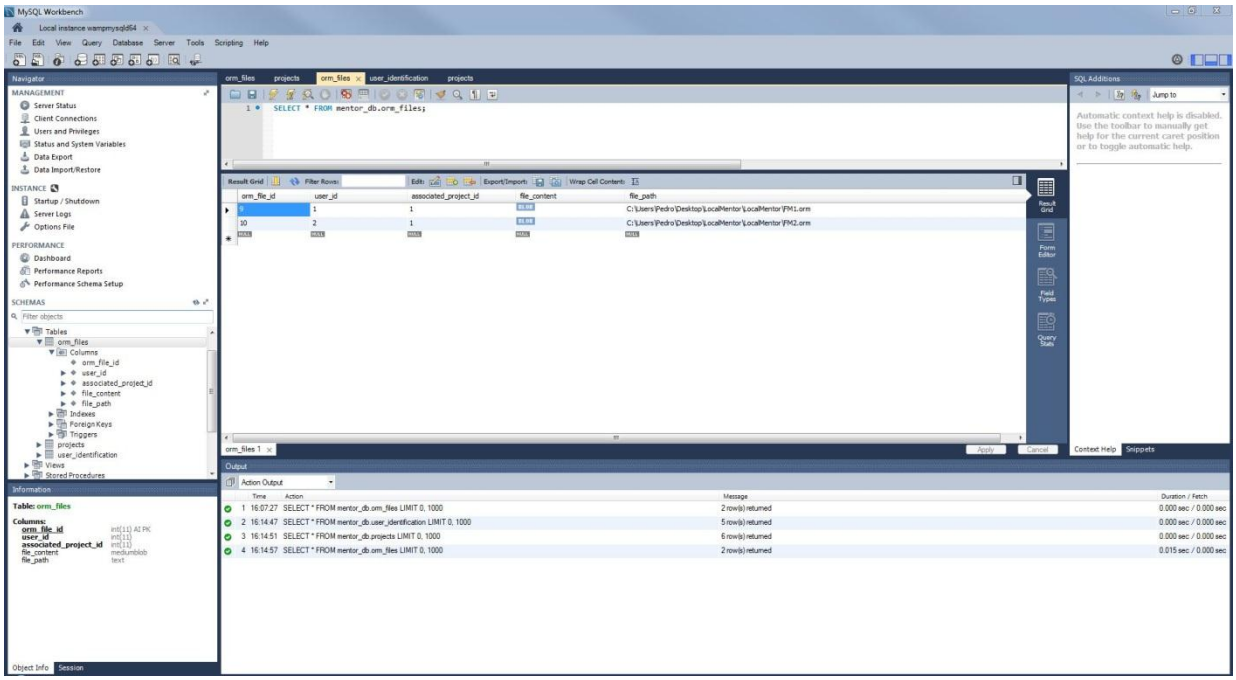


Fig. 4.3: Projects Table of the Project/Users DB

4.3.2 Mismatches Mediator Data Base

Like the previous Data Base, this DB was built using the MySQL tool and serves has a way of storing the gathered mismatches. For that to be possible, the DB was constructed with the structure of the already presented Mediator Ontology (Fig. 3.5). This allows the user to check the mismatch information before passing it to the Mediator Ontology itself and allows to manually altering the mismatch information.

4.3.3 MENTOR User Interface using Visual Studio

The User Interface was developed in the C# programming language, mainly because the NORMA plugin is only available for the .NET Framework. There is no theoretical reason that NORMA couldn't

be used in another object-oriented language (like Java), but the NORMA plugin takes advantage of certain aspects of the C# language, that are exclusive to it. Some of these aspects are:

- C# has LINQ, a very handy way that the Microsoft Developers provided in order to express queries directly in C#. This formation of these queries don't rely on the implementation details of the thing being queried, allowing the creation of queries for databases, in-memory objects and XML.
- The generated .NET code makes extensive use of delegates and generics, which are not supported by other languages, like Java;
- Since the NORMA tool was created with the additional intent of allowing ORM models to be mapped to other types, like relational schemes, databases and object oriented languages, all of the NORMA code generators go through a PLiX (programming Language in XML) generation framework, generating XML representations of the code first and then formatting to text. This ability isn't provided by other object oriented languages.

The developed C# libraries are responsible for managing the execution of the reference lexicon settlement phase of the MENTOR methodology adaptation, providing user interface interaction.

4.3.4 Mediator Ontology Tool

The Mediator Ontology Tool was already fully described in the subsection 3.2.1. This tool was developed on the Java programming language, being adapted by the author in order to be able to receive mismatch information from Fact Models.

4.4 Detailed Process

In this subsection, a detailed description of the developed methodology is presented for a better understanding of how the Fact Models can contribute for the formalization and conceptualization of the business domains, allowing an overall understating of the lexicon settlement process. To illustrate each of the steps that compose this process, the author will present a flow chart that describes each of these steps.

4.4.1 Domain Definition and Terminology Gathering Step

Before starting to extract knowledge and initiate the lexicon settlement phase, a project needs to be created. In order for that to happen, one has to first connect to the user interface. The user interface allows to:

- Add new users;

- Remove another user (If connected as admin);
- Add new projects;
- Remove a project (if connected as admin);
- Associate new Fact Models (ORM files) to an existing project;
- Remove Fact Models from the project;
- Update a Fact Model inside a project;
- Start and finish each of the steps for the lexicon settlement.

After a project has been created and as soon as all the necessary Fact Models have been submitted to that project, the administrator can then start the first step of the lexicon settlement phase. The fig. 4.4 displays a flow chart that illustrates every single detail of the Terminology Gathering Step.

First, the user must connect to the user interface as the administrator, since clients are only allowed to create a new project or to contribute to the reference lexicon settlement by adding a new ORM file to a specific project.

As soon as the user connects to the user interface as an administrator, the system checks the User/Projects Database for the existence of projects and checks if there are any projects that are able to be part of the reference lexicon settlement, that is, the system only allows projects that have more than one ORM file associated, since it wouldn't make any sense to settle a reference lexicon with a single business model.

When the admin selects a valid project, the knowledge extraction begins. This step consists on a cycle that goes through all the ORM Files that are associated to the chosen project, extracting every single Terms and Fact Types that represent the business domain expressed by the Fact Models. The extracted information includes properties of the Terms and the Fact Type, such as the mandatory and uniqueness constraints, note descriptions written by the each model proprietary, type of multiplicity, similar to the UML notation, etc.

Once all the information has been retrieved, the admin can now review the extracted terms and fact types from all the contributed Fact Models, by selecting one by one and checking all the relevant information of each of these items with the aid of the verbalization browser tool, embedded in the developed methodology.

When the admin finished the Term and Fact Type revision, the methodology can proceed to the next step of the methodology, the Glossary Building Step.

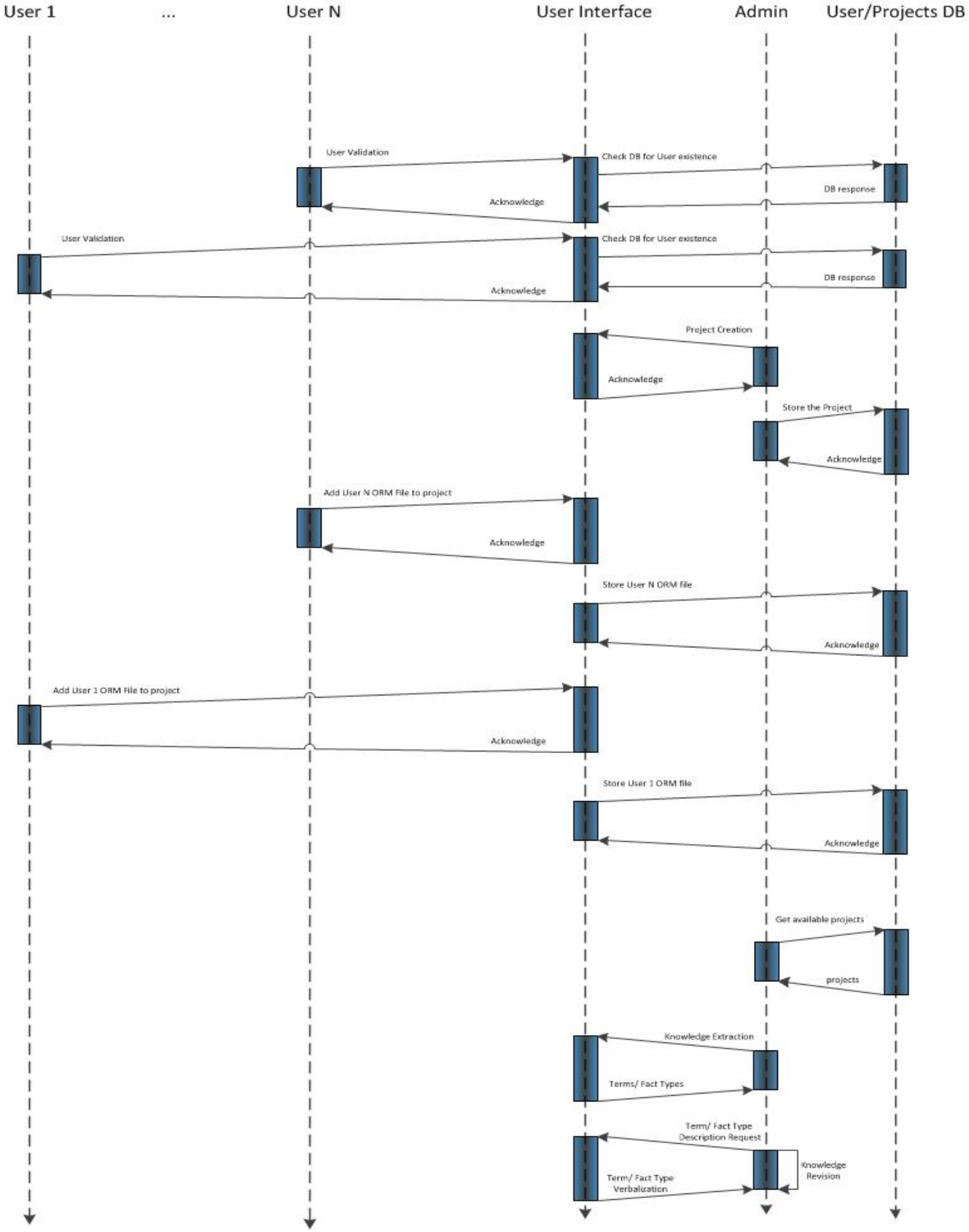


Fig. 4.4: Terminology Gathering Step Flow Chart

4.4.2 Glossary Building and Mismatch Detection Step

In this step, the admin user has the task to select the proper Terms and respective definitions that will be part of the reference glossary. A representation of the Glossary Building and Mismatch Detection Step is illustrated in fig. 4.5.

The admin starts by selecting each one of the terms extracted, activating a 2-phase process algorithm developed for detecting possible term mismatches. First, the algorithm checks if the selected term in the model A is present in the model B by using simple relations like the ones already mentioned in section 4.2.2 (syntactically equal terms/descriptions).

The second phase of this mismatch detection process consists in the detection of terms synonyms, using the myThes tool provided by the NHunspell API.

Once this process finish for each of the terms, the selected term and its possible mismatch terms are displayed in the User Interface along with their respective descriptions. These descriptions can either be provided by the proprietary models, as note descriptions, or be obtained by obtaining the fact types and constraints that are related to the selected term, forming a query with all the conceptual information exposed by the model.

The user then needs to select the term and the respective description that is most appropriated for the future reference model. For that, the user can either selected one of the displayed terms and descriptions, or give a new name and/or new description. In addition, the user may find the term inappropriate to the reference model, it might be needless, so the user has the ability to simply ignore the selected term and continue the cycle for the next term.

Every time a term is selected, a semantic mismatch is established, providing a link between the proprietary terms to the reference terms. In this case, the term that the user chooses will be a reference term, and so, that term will be linked to all the terms that were detected as semantic mismatches.

Once the admin has selected all the reference terms, the semantic mismatch mappings are stored to the Mediator Mismatches Database and transferred to the mediator ontology using the RESTful WebService and the glossary is stored in a Database so that the business enterprise can consult it for business domain reference.

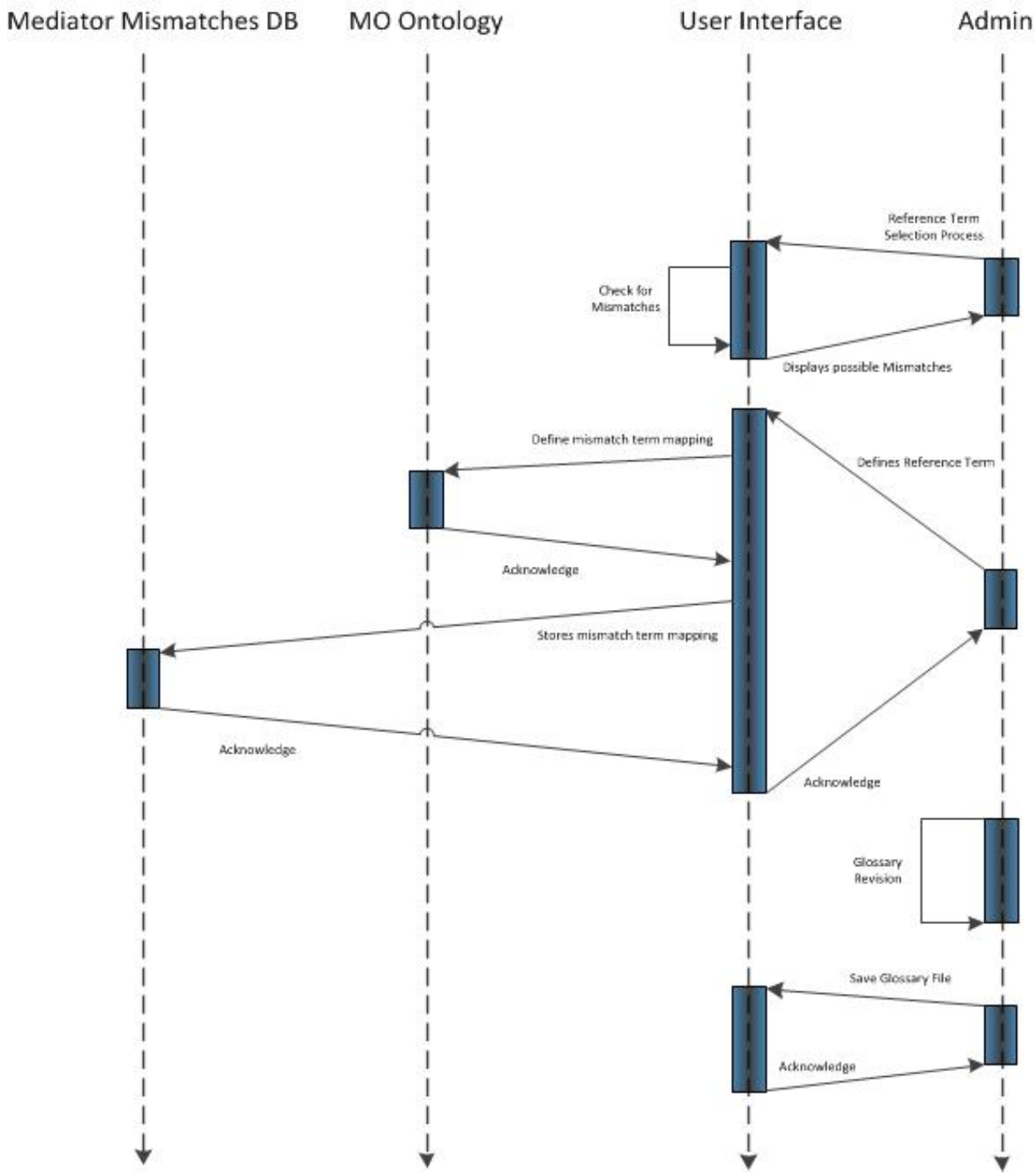


Fig. 4.5: Glossary Building Step Flow Chart

4.4.3 Thesaurus Building Step

Once the Business Glossary is defined and stored, the administrator can then continue to the next step of the methodology, the step where taxonomic relationships between each of the business terms are defined, concluding the Reference Lexicon Settlement phase of the methodology.

Knowing that taxonomies follow the same type of relationship to each of its terms, the administrator must first decide the type of thesaurus that will be built. That said, in order for a Thesaurus to be built, the administrator needs to take into account that the methodology is using Fact Models for domain expression, and so, the type of relationships we are looking for, are in the form of Fact Types. So, the administrator must pick a verb phrase or preposition that represents a Fact Type connection.

As soon as the administrator picks the type of Thesaurus that it is intended, an automatic tree structured is displayed by the methodology in the user interface. The automation of this process is possible because the methodology analyses all the terms in the fact model that share the picked fact type notation, groups them and goes through a cycle to define which one is the most general node (father node) and which ones are the branches (children nodes).

4.4.4 Other Steps

After accomplishing the Thesaurus building step, the reference lexicon is settled. From here, the author has the ability to export the mismatch data from the local database to the MO methodology, or directly export this data from the developed methodology in MS Visual Studio to the MO methodology, using the RESTful web service. When the MO methodology is loaded with all the mismatch information, each of the stakeholders can start communicating with each other by sending messages to the mediator, which will translate those messages to the other stakeholder format.

4.5 Concluding Remarks

In this chapter, the author presented the purposed methodology architecture, explaining each of the developed steps in detail and introducing the used tools for that matter. The study conducted in this chapter enabled a better understanding of the system and how all the components interact with each other and with the aid of the flow charts presented in section 4.4, a visual and temporal understanding of how the reference lexicon settlement and semantic mapping representations is fulfilled is simplified for the reader and used as further reference during the development of the methodology.

5

Demonstrator Testing and Hypothesis Validation

The architecture presented on the previous chapter was implemented according to all established parameters, and its results are shown in this chapter. For this chapter, the built methology will be put into test using a practical example in a business environment, proving in that way, the hypothesis purposed in the subsection 1.5 of the first chapter of this dissertation.

5.1 Methodology Developing Demonstration

For the demonstration of the developed methodology, a practical example inside an enterprise environment is presented. This example represents a simple Database enterprise structure design used to register all the enterprise employees and their respective personal information.

Before connecting to the methodology user interface, each contributor must define its business domain structure, which in this case, is the proprietary Fact Model. Fig.5.1 and 5.2 illustrate the contributed Fact Models for this simple demonstration.

The first Fact Model contains 4 entity types, terms that are connected to a value identifier, the reference scheme and 8 value types, constant value terms. To keep this experimentation simple, the author decided to only include simple constraints, such as, mandatory constraints, the constraints that are used to define the absolute essential fact types necessary for the business domain to function properly, and uniqueness constraints, used to define the kind of multiplicity shared by each of the fact types.

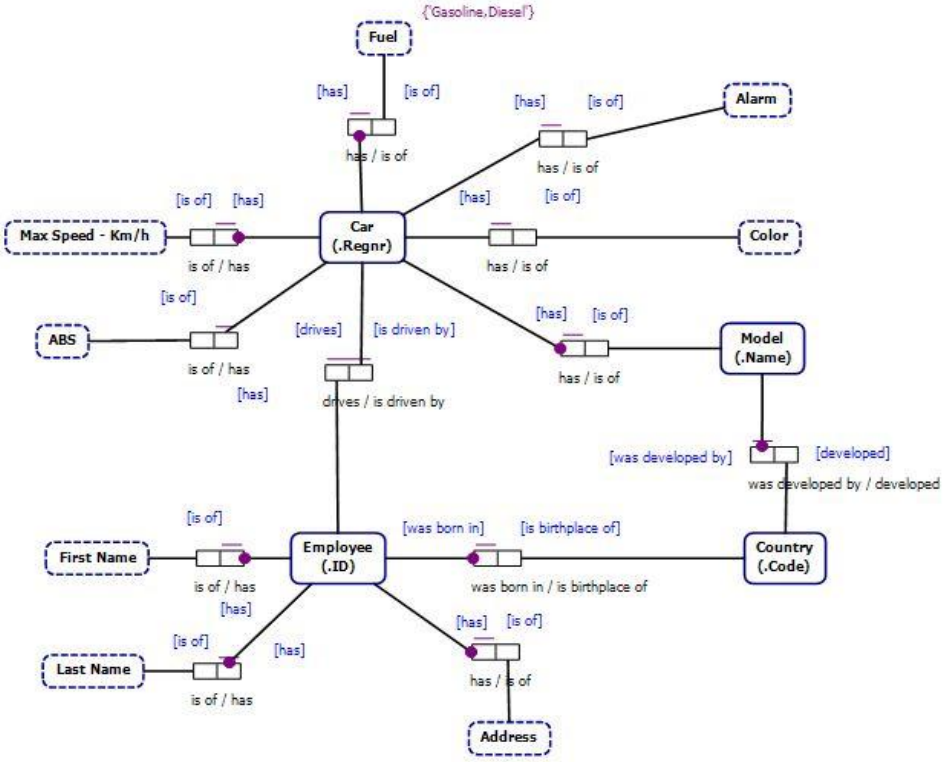


Fig. 5.1: Fact Model provided by the enterprise A

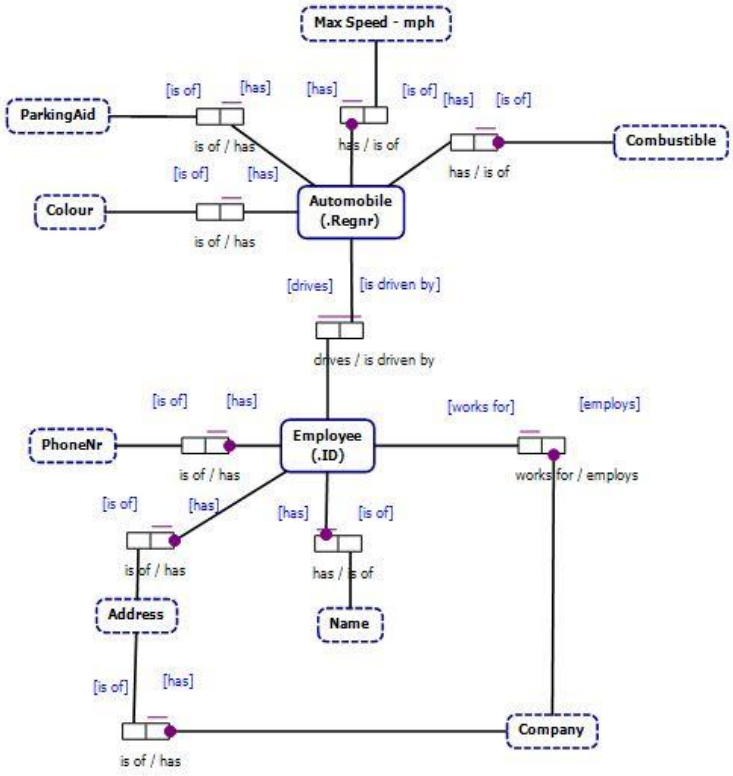


Fig. 5.2: Fact Model provided by the enterprise B

The idea behind this simple example, was to represent two very similar business structures, inside the same type of business domain, but with some slightly differences in order to provide possible semantic mismatches. The type of semantic mismatches that will be covered are, naming, granularity, structuring, encoding, content and coverage mismatches. These types of mismatches will be further explained with more detail.

When each of the business partners first connects to the methodology local user interface, they will be prompt with a Login Form where, if already registered, they can access to their private area. If they are not registered, it is possible to create a new user by adding all the necessary information to the system. The Login Form can be seen on Fig. 5.3:

Fig. 5.3: User Interface Login Form

In order to connect to the developed methodology, first the user must select the intended login type (connect as a simple user or as an administrator).

As the user clicks on the submit button, the methodology starts communicating with the MySQL Database for user verification. In case of being a new user, the user can opt to click on the sign up button, so that the insertion of a new user, as a simple user or an administrator, on the Data Base can occur.

The Fig 5.4 presents the project form. When a user connects to this area, the developed methodology starts by connecting to the Database, gathering all the available projects and displaying then. In this area the users can contribute with their business domains, that is, their Fact Models, associating their Fact Models to the chosen project. Similarly, the user can add new projects to the methodology, storing then on the MySQL Database.

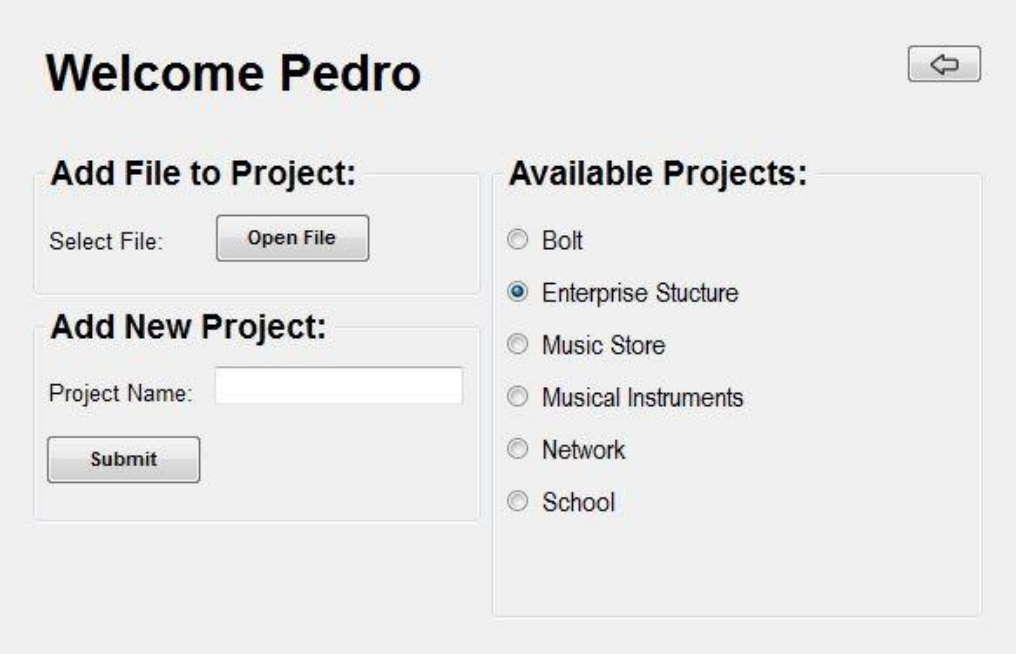


Fig. 5.4: User Interface Project Form

After all the users have contributed with their own business domain structures, the administrator connects to the admin area. When the administrator first enters in this area, the methodology starts by connecting once more to the Database, gathering all the available projects. This time, a query was used to gather all the associated users to the respective projects. In the Fig. 5.5 is possible to observe that the methodology defined the project "Enterprise Structure" in green and the rest in red. This means that the project "Enterprise Structures" has two or more Fact Models associated, which means that the project is valid for the knowledge extraction step.

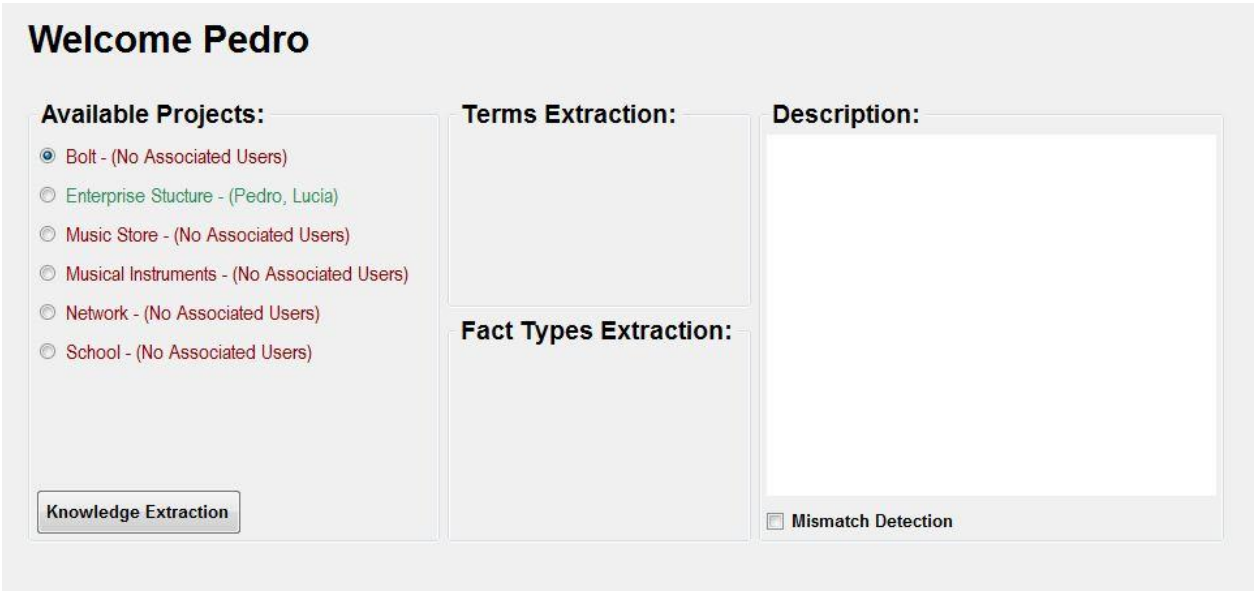


Fig. 5.5: Admin User Interface

Once the admin selects the intended project and clicks on the "Knowledge Extraction" button, the methodology starts by analysing all the Fact Model ORM files that were associated to the project and using the NORMA API libraries, starts extracting all the terms and fact types that form them and merge all the terms in a single list of terms and all the fact types in a single fact type list. The developed methodology then fills dynamically the combo boxes with all the terms and fact types.

Now the administrator can review all the data that was extracted by selecting one of the terms or fact types on the combo boxes by checking the imported verbalization browser on the right. In Fig 5.6, the admin selected the term "Car", revealing all the information related to this term, which includes, term type, reference scheme, fact types associated to and possible note descriptions.

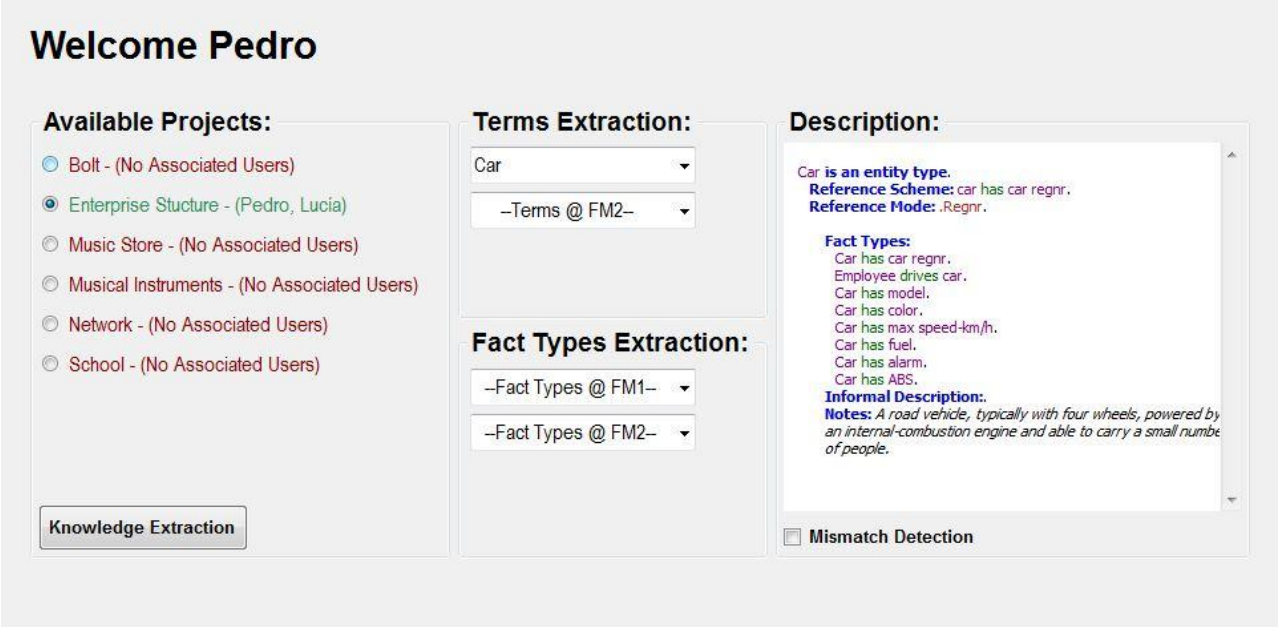


Fig. 5.6: Term Verbalization and Revision

Now that all the content from the Fact Model ORM files have been extracted, the user can start the selection of the terms and respective description that will be part of the reference lexicon. For that, the user needs to simply check the "Mismatch Detection" checkbox before selecting a term in the combo box.

When the administrator does that, the methodology begins the process of finding possible term synonyms of the selected term. The methodology goes through two processes as already mentioned in subsection 4.2.2 and subsection 4.4.2. First, the methodology tries to find in the other models, terms with:

- Same term name and same term description. If this is the case, the methodology simply eliminates one of the two from the merged list and defines the other as the reference;

- Same term name but different term description. In this case the methodology presents only one of the term names and presents to the administrator the two possible descriptions for that term;
- Different term name but equal term description. Oppositely to the last case, the methodology displays the two possible term names but only shows one possible term description, since the two are the same;
- Different term name and different term description. In this case, the administrator has to choose the term name and the term description that will be part of the reference lexicon.

The second process consists on using the libraries of NHunspell to check for possible synonyms of the selected term name. The developed methodology checks, using the MyThes tool of NHunspell if any of the terms of the other Fact Models is a synonym of the selected term. If so, the term and description of that term is displayed.

Finally, the methodology displays other type of term descriptions, using the fact type relations of the selected terms. In the example show in Fig. 5.7 When the administrator selects the term "Car" the methodology detects a synonym "Automobile" and the methodology displays three possible descriptions, a note description of the term "Car", a description based on fact types of the term "Car" and a description based on fact types of the term "Automobile". Since the note description of the term "Automobile", was equal to the note description present in the term "Car", the methodology simply ignores it.

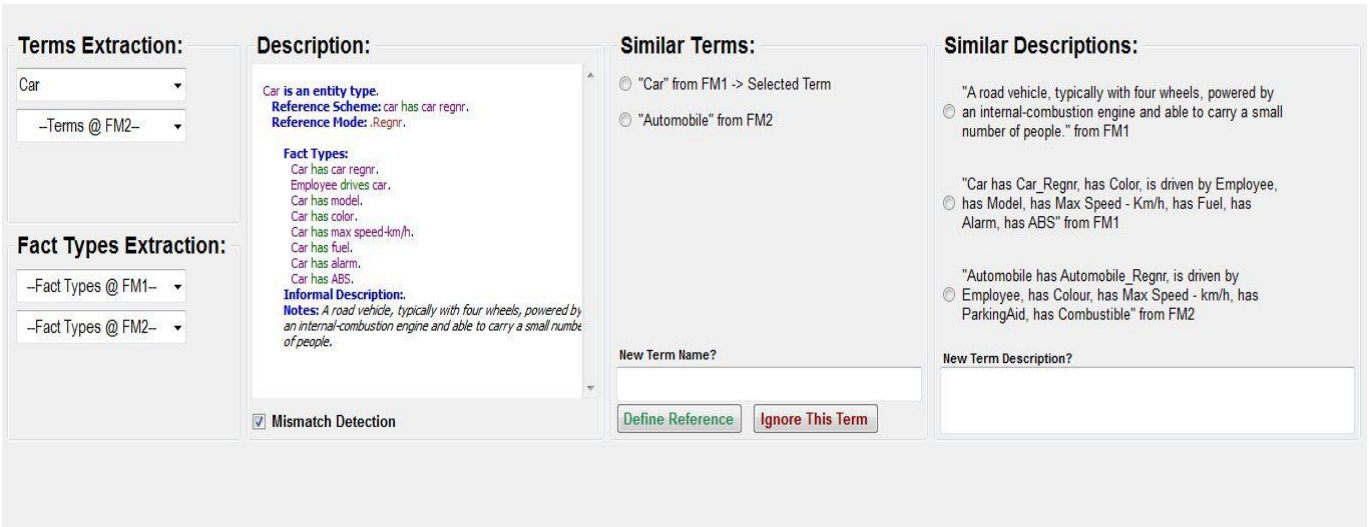


Fig. 5.7: Mismatch Detection and Display

Now the administrator has to choose the proper term and description. For that, the administrator can either select one of the displayed terms and descriptions or write a new one. If the administrator finds that the selected term will be unnecessary to the reference lexicon, it can simply be ignored by clicking on the "Ignore This Term" button.

When the administrator decides which will be the term that will be part of the reference lexicon, the "Define Reference" button must be clicked. As soon as the button is clicked, the chosen term and description are defined as reference and are added to the glossary. In the case of the not selected term names, these are defined as mismatches, linking the chosen term that is now part of the reference model and the mismatch terms on each of the proprietary models.

Another thing that is worth of mentioning is that, when the administrator defines a term as reference, that term is removed from the combo box where it was before being selected, in order to avoid further redundancies. The same happens to all the related terms, synonyms and reference schemes. So, if we were to define the term "Car" as a reference term, the terms "Automobile"," Car_regnr" and "Automobile_regnr" would have been removed to. The Following figure illustrates the combo box of the Enterprise's A and Enterprise's B Fact Models, after having selected the term "Car" has a reference term.

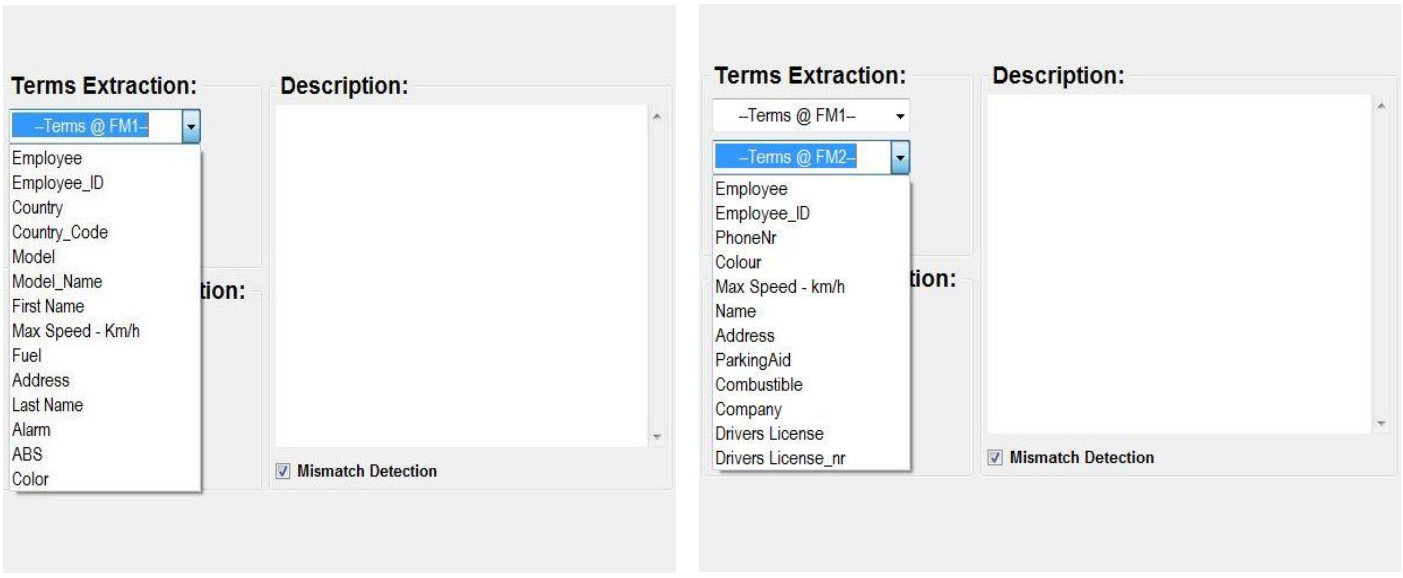


Fig. 5.8: After selecting the "Car" term as reference

A particular example of mismatch detection is the encoding type mismatch, which detects equal terms that have different but equivalent units. An example of this is the use of the metric system by most countries in the world and the imperial system used by the United Kingdom. The Following figure exhibits the detection of this kind of mismatch by the developed methodology, as it detects the term "Max Speed - Km/h" and possible equivalent term "Max Speed - mph".

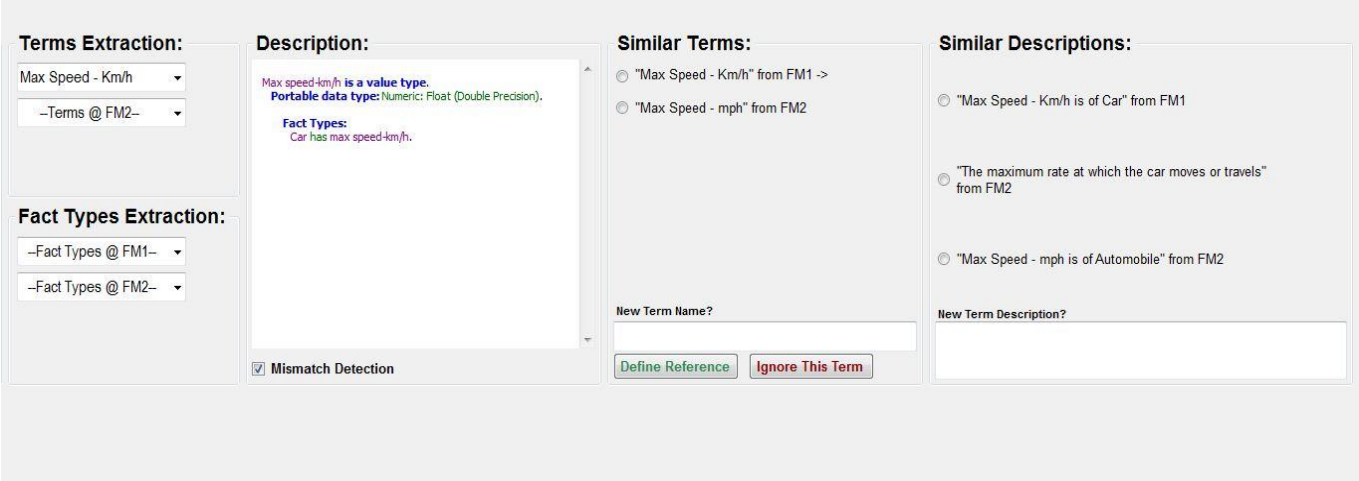


Fig. 5.9: Encoding Mismatch Example

After all the reference terms and respective descriptions have been selected by the administrator, these are displayed so that it is possible to review and change them. In the table 5.1, it is possible to see the obtained alphabetic list of terms that the administrator has chosen during this procedure. Once the administrator is satisfied with the collected terms and descriptions, the business glossary is saved on the MySQL data base for future reference.

As the Glossary is stored in the MySQL data base, the mismatches that were detected during the glossary building process are prepared and saved on a dedicated data base table.

This table is presented in table 5.2. As it is possible to observe, the mismatches Naming, Content, Granularity, Equal and Encoding were covered for this example. Of course, it would be possible to use many more types of mismatches, but these are enough for this dissertation purpose. The table is organized by the following headers:

- **Proprietary Term Name:** Refers to the tested term from the contributed Fact Models;
- **FM Contributor:** The contributor identifier. This allows the methodology to know which of the participants contributed a certain term;
- **Reference Term Name:** This is the term that the administrator decided add to the reference lexicon. This term combined with the proprietary term name makes the Melems pair (A,B);
- **KMType:** As already mentioned, this is the Knowledge Mapping Type. In this example the only mapping types that were tested were the Structural and the Conceptual Type.

- **Match Class:** This is the Match/ Mismatch Classification;
- **Exp:** Finally, this the expression used to classify and help the mediator on defining the relationships between the proprietary term names and the reference terms. In this case, the author has used two expressions for each interaction. The author decided to do this so that, besides defining the relationship between a proprietary term name and a reference term, the mediator would have the ability to know the relationship between the involved proprietary terms. A practical example of this is the term "Alarm". In this example, this term was decided to be out of the reference lexicon, so the author used the "Ignore This Term" button to ignore it. The effect of this action in the mismatch table is to indicate that the Fact Model A (provider of the term "Alarm"), has content that the reference model does not (A ⊇ Ref). The same goes for the Fact Model B (A ⊇ B). This is defined as a "Structural Content Mismatch", since the reference model doesn't have this fact type branch.

Reference Term	Term Description
ABS	ABS is of Car
Address	The number of a building and the name of the street
Automobile	A road vehicle, typically with four wheels, powered by an internal-combustion engine and able to carry a small number of people. Automobile is identified by Automobile_Regnr
Colour	The quality of an object or substance with respect to light reflected by the object, usually determined visually by measurement of hue, saturation, and brightness of the reflected light
Combustible	Type of substance that can be burned as a source of power for the engine
Company	Company employs Employee, has Address
Country	A Country has Country_Code, is birthplace of Employee, developed Model, A Country is identified by Country_Code
Employee	Employee has Employee_ID, drives Car, was born in Country, has First Name, and has Last Name. Employee is identified by Employee_ID
Max Speed - mph	The maximum rate at which the car moves or travels
Model	Model has Model_Name, is of Car, and was developed by Country. A Model is identified by a Model_Name
Name	A word or set of words by which a person or thing is known
ParkingAid	A set of sensors localized on the rear bumper of the Automobile used to alert the driver from obstacle proximity
PhoneNr	Fixed set of numbers used as an identifier or specific telephone

Table 5.1: Reference Glossary

Proprietary Term Name	FM Contributer	Reference Term Name	KMType	MatchClass	EXP
ABS	FM A	ABS	Conceptual	Equal	A = Ref / A ⊇ B
Address	FM A	Address	Conceptual	Equal	A = Ref / A = B
Address	FM B	Address	Conceptual	Equal	B = Ref / A = B
Alarm	FM A	::: Unused :::	Structural	Content	A ⊇ Ref / A ⊇ B
Automobile	FM B	Automobile	Conceptual	Equal	B = Ref / A = B
Automobile_Regnr	FM B	Automobile_Regnr	Conceptual	Equal	B = Ref / A = B
Car	FM A	Automobile	Conceptual	Naming	A = Ref / A = B
Car_Regnr	FM A	Automobile_Regnr	Conceptual	Naming	A = Ref / A = B
Color	FM A	Colour	Conceptual	Naming	A = Ref / A = B
Colour	FM B	Colour	Conceptual	Equal	B = Ref / A = B
Combustible	FM B	Combustible	Conceptual	Equal	B = Ref / A = B
Company	FM B	Company	Conceptual	Equal	B = Ref / A ⊆ B
Country	FM A	Country	Conceptual	Equal	A = Ref / A ⊇ B
Country_Code	FM A	Country_Code	Conceptual	Equal	A = Ref / A ⊇ B
Employee	FM A	Employee	Conceptual	Equal	A = Ref / A = B
Employee	FM A	Employee	Conceptual	Equal	A = Ref / A = B
Employee_ID	FM B	Employee_ID	Conceptual	Equal	B = Ref / A = B
Employee_ID	FM B	Employee_ID	Conceptual	Equal	B = Ref / A = B
First Name	FM A	Name	Structural	Granularity	A ⊆ Ref / A ⊆ B
Fuel	FM A	Combustible	Conceptual	Naming	A = Ref / A = B
Last Name	FM A	Name	Conceptual	Granularity	A ⊆ Ref / A ⊆ B
Max Speed - Km/h	FM A	Max Speed - mph	Structural	Encoding	A = Ref / A = B
Max Speed - mph	FM B	Max Speed - mph	Conceptual	Equal	B = Ref / A = B
Model	FM A	Model	Conceptual	Equal	A = Ref / A ⊇ B
Model_Name	FM A	Model_Name	Conceptual	Equal	A = Ref / A ⊇ B
Name	FM B	Name	Structural	Granularity	B = Ref / A ⊆ B
Parking Aid	FM B	Parking Aid	Conceptual	Equal	B = Ref / A ⊆ B
PhoneNr	FM B	PhoneNr	Conceptual	Equal	B = Ref / A ⊆ B

Table 5.2: Obtained Mismatch table

After storing this mismatch data in the data base, this information can be transferred to the mediator ontology. This is where the RESTful API gracefully enters, providing web services to transfer this information by small chunks, mismatch by mismatch pair.

Now that the glossary has been accomplished, the administrator can go forward to the next step, the Thesaurus Building Step. In this step, the author has to choose the type of thesaurus is intended to be built. As already mention in the subsection 4.4.3, the type of the thesaurus is defined by the fact types that connect the involved terms in the reference lexicon. For instance, for this example, the administrator chose to build a "has" type thesaurus, so the thesaurus can only have "has" relationships between each tree node like in the following figure:

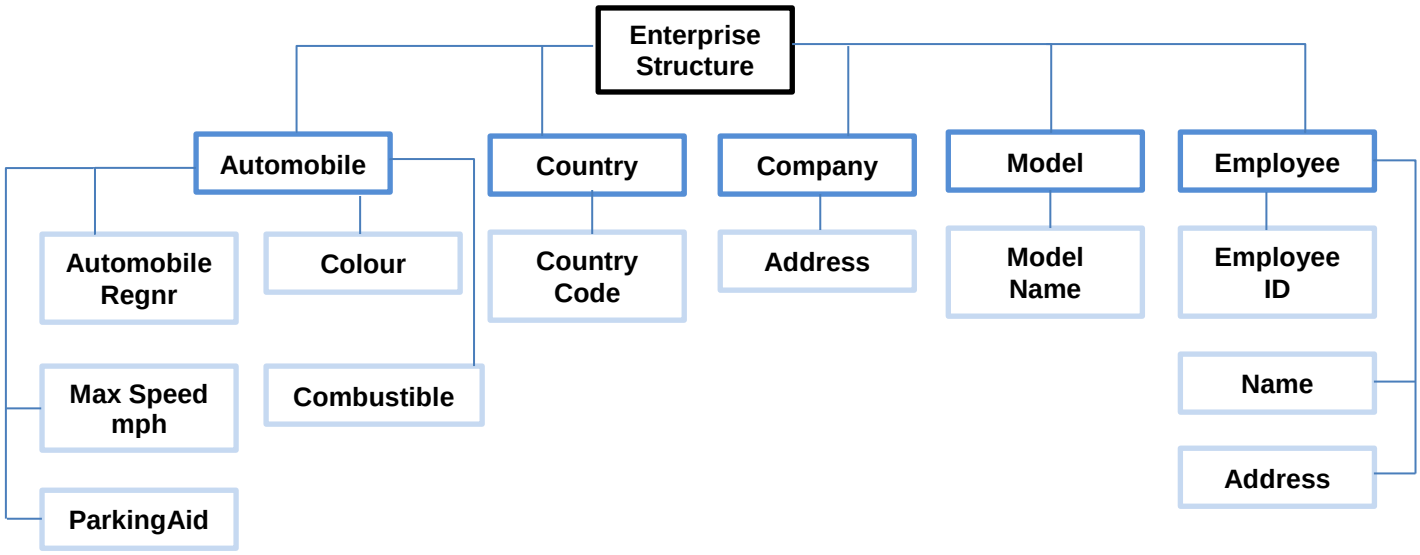


Fig. 5.10: Expected "has" Thesaurus Structure

As soon as the administrator types the "has" verb phrase and clicks the "Define Thesaurus" button, an automatic tree structured is displayed by the methodology in the user interface. The automation of this process is possible because the methodology analyses all the terms in the fact model that share the picked fact type notation, groups them and goes through a cycle to define which one is the most general node (father node) and which ones are the branches (children nodes). The following figure illustrates the Tree View Thesaurus obtained by the administrator:

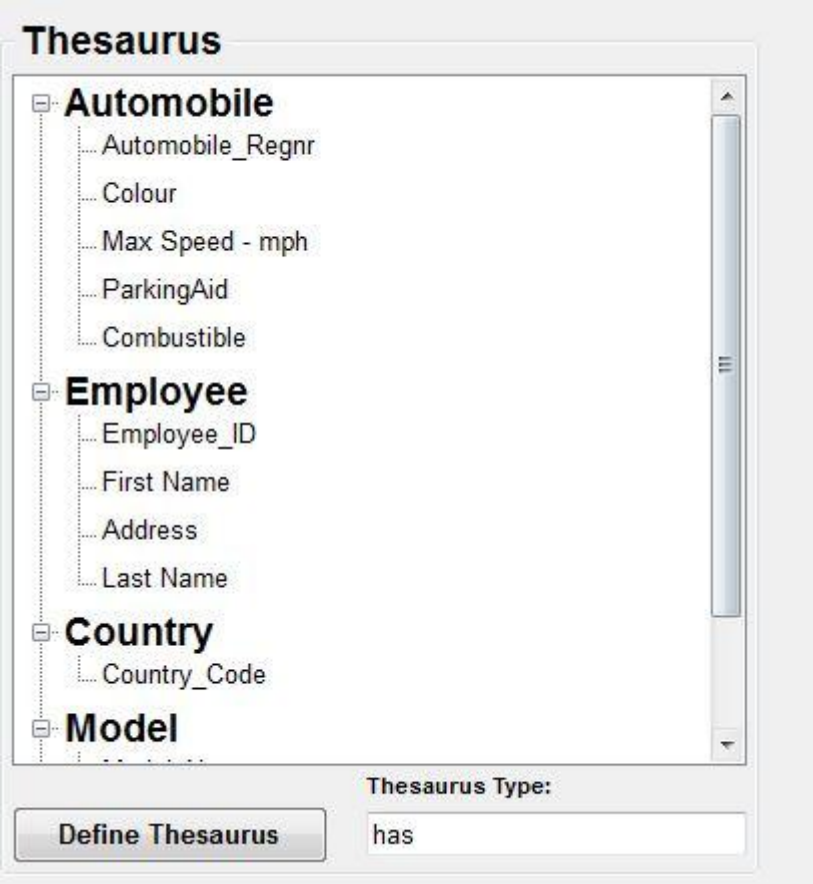


Fig. 5.11: Obtained "has" Thesaurus

As it is possible to observed, the obtained Thesaurus structure is exactly as it was expected to. With the Thesaurus defined, the first phase of this methodology (reference lexicon settlement) comes to an end.

5.2 Dissemination Executed and Hypothesis Validation

In respect to the hypothesis validation, the author demonstrated in the chapter 5, section 5.1, that by designing a methodological approach for building a domain of reference lexicon based on a methodology for reference ontology building (MENTOR), which accomplishes semantic mapping tables, that it was possible to building a reference domain of discourse that serves as a middle-man between the involved stakeholders. With this methodology, the participating users were able to build a single business structure using parts of each of the proprietary business models added to the project.

The author was able to use Fact Models to express a business domain, allowing a conceptualized view; the extraction of data from the Fact Models and further manipulation of these by using external libraries; able to build a glossary with the reference terms and respective descriptions, defining a first set of terms mapping (semantic mismatches); able to develop an algorithm to automatically build a

taxonomic structure (thesaurus) using the gathered terminology and the relations between each of these terms (fact types).

With this methodology, it is possible to create a common and understandable business domain that serves as a connection between enterprises that originally didn't share the same domain of discourse, promoting business interoperability.

6

Conclusion

It is no lie saying that today's business market is an ever-growing machine, every day a bit more demanding and rigorous. In consequence to this, the SME are the ones that first suffer with this continuous growth, getting behind the apparently unreachable Big Enterprises.

In order to turn the odds in favour to these SMEs, or at least allowing them to keep in business and competitive, collaboration agreements between these enterprises were necessary, provoking an explosion on the development of methodologies and technologies able to bring these enterprises more closer to each other. Of course this task wasn't as easy as it sounds. Due to the fact that even in a same community or domain, there were a big variety of knowledge representation elements, many interoperability problems have been identified.

This is where methodologies, such as the MENTOR, come into action. The methodology appears with the idea of creating a common representation of the knowledge shared by a set of enterprises, providing a middle-man action between the involved domain structures and lexicons and allowing these enterprises to keep their information models while the methodology creates a reference information model with the common understanding of the domain of discourse.

Besides the interoperability problems, stakeholders have reported to have a high difficulty in understanding and keeping in track with the data models design and creation, having no choice but to ask the data modeller engineer, responsible for the construction of the data models, to clarify them about certain aspects of the models. To attend this problem, data conceptualization was needed.

In this dissertation a solution was implemented to take advantage of the MENTOR methodology, providing an adaptation of this same methodology. The methodology adaptation provided an establishment of an enterprise reference lexicon from business data models, addressing the automation on the Thesaurus building step and the conceptualization and formalization of the business domain, with a clear definition of the used lexicon to facilitate an overall understanding by all the involved business stakeholders.

6.1 Future Work and Propositions

Although the fact that the hypothesis of this dissertation was a success, there is much more work that can be done, and much more space for refinements. For future work, it would be important to study with more precision and more extensively all the available constraints and characteristics of the Fact Models, taking advantage of these types of constraints when passing them to the mediator ontology. It would be equally important to build an adaptation of MENTOR where the model would receive models of multiple types (Fact Model and an Ontology for instance), and not just a single model type, and build a reference between these multiple types. Another important aspect is ontological exportation to the protégé tool. This dissertation only covered the methodology steps until the mediator ontology. However, in order to pass this information to the protégé tool, one would need to convert this information to an ontological format.

Finally, knowing that the Thesaurus can be built with another type of ontological relations, it would be important to explore this functionality in order to build various conceptual structures of the same model that would facilitate the next phase of the methodology, and perhaps, promote the automation of these steps.

7

References

[1] J. Sarraipa, R. Jardim-Goncalves, T. Gaspar, and A. Steiger-Garcao, “Collaborative ontology building using qualitative information collection methods,” *2010 5th IEEE Int. Conf. Intell. Syst.*, pp. 61–66, Jul. 2010.

[2] G. F. P. L. Alves, “A Framework for Semantic Checking of Information Systems,” 2012.

[3] G. Witt, *Writing Effective Business Rules*, vol. 4. Elsevier, 2012, 2012, p. 360.

[4] L. M. Camarinha-matos and B. Terminology, “SCIENTIFIC RESEARCH Unit 2 : SCIENTIFIC METHOD,” pp. 2009–2012, 2012.

[5] S. C. Shapiro, “Knowledge Representation and Reasoning Logics for Artificial Intelligence,” 2010.

[6] S. A. A. Mappe, “Knowledge Representation for Potential Field of Study Recognition,” no. August, pp. 160–163, 2013.

[7] C. Lucena, U. N. De Lisboa, S. Baldiris, R. Fabregat, and S. Aciar, “The ALTER-NATIVA book for Knowledge Representation.”

[8] S. Dietzold, J. Lehmann, and T. Riechert, “OntoWiki A Tool for Social , Semantic Collaboration Categories and Subject Descriptors,” 2007.

[9] A. Fatwanto, “Specifying translatable software requirements using constrained natural language,” in *2012 7th International Conference on Computer Science & Education (ICCSE)*, 2012, no. Iccse, pp. 1047–1052.

[10] J. Lyons, *Natural language and universal grammar*. Cambridge: Cambridge University Press, 1991.

[11] F. Fabbrini, M. Fusani, V. Gervasi, S. Gnesi, and S. Ruggieri, “Achieving Quality in Natural Language Requirements,” in *Proceedings. 11th International Software Quality Week, San Francisco*, 1998.

[12] H. Lethanh, “6. Natural language processing.,” *Stud. Health Technol. Inform.*, vol. 205, p. 563, Jan. 2014.

[13] H. M. Harmain and R. Gaizauskas, “CM-Builder: an automated NL-based CASE tool,” in *Proceedings ASE 2000. Fifteenth IEEE International Conference on Automated Software Engineering*, 2000, pp. 45–53.

[14] S. P. Overmyer, L. Benoit, and R. Owen, “Conceptual modeling through linguistic analysis using LIDA,” in *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*, 2001, pp. 401–410.

[15] D. Hutchison and J. C. Mitchell, “Conceptual Model Generation from Requirements Model: A Natural Language Processing Approach,” in *13th International Conference on Applications of Natural Language to Information Systems, NLDB 2008, London, UK, June 2007, Proceedings*, 2008, p. 270.

[16] R. Schwitter, “English as a formal specification language,” in *Proceedings. 13th International Workshop on Database and Expert Systems Applications*, 2002, pp. 228–232.

[17] E. Yahia, A. Aubry, and H. Panetto, *Computers and Translation*, vol. 35. Amsterdam: John Benjamins Publishing Company, 2003, pp. 1–33.

[18] O. M. Group, “Business Semantics of Business Rules,” pp. 1–50, 2003.

[19] O. M. Group, “Semantics of Business Vocabulary and Business Rules,” no. January, 2008.

[20] I. S. Bajwa, B. Bordbar, and M. G. Lee, “OCL Constraints Generation from Natural Language Specification,” *2010 14th IEEE Int. Enterp. Distrib. Object Comput. Conf.*, pp. 204–213, Oct. 2010.

[21] I. S. Bajwa, M. G. Lee, and B. Bordbar, “SBVR Business Rules Generation from Natural Language Specification,” pp. 2–8, 2011.

[22] S. Spreeuwenberg and K. A. Healy, *SBVR’s Approach to Controlled Natural Language*, vol. 5972. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[23] D. Hay, K. A. Healy, J. Hall, C. Bachman, J. Breal, J. Funk, J. Healy, D. Mcbride, R. Mckee, T. Moriarty, L. Nadeau, and S. Quarles, “Defining Business Rules: What Are They Really?,” *Bus. Rules Gr.*, 2000.

[24] R. G. Ross, *Business Rule Concepts: Getting to the Point of Knowledge*, 4th ed. Business Rule Solutions, LLC, 2013.

[25] M. Bajec and M. Krisper, “A methodology and tool support for managing business rules in organisations,” *Inf. Syst.*, vol. 30, no. 6, pp. 423–443, Sep. 2005.

[26] E. Bauer, “The Business Rule Approach,” 2009.

[27] R. G. Ross, *Principles of the Business Rule Approach*. Addison-Wesley Professional, 2003.

[28] T. Morgan, *Business Rules and Information Systems : Aligning IT with Business Goals*. Addison-Wesley Professional, 2002.

[29] J. Purchase, *What is a Fact Model? Why Should You Have One?* available from http://blog.luxmagi.com/2010/04/what-is-a-fact-model-why-should-you-have-one/, 2010.

[30] M. A. Poolet, “Visualizing Business Rules.” available from http://sqlmag.com/business-intelligence/visualizing-business-rules, 2007.

[31] G. Witt, “Developing a Fact Model to support SBVR - Compliant Rule Statements,” 2012.

[32] T. Halpin, “Object-Role Modeling : an overview,” pp. 1–15, 2001.

[33] “Fact Based Modeling WG.” available from http://www.factbasedmodeling.org/, 2011.

[34] O. M. O. R. M. Niam and T. Halpin, “Data modeling in ORM,” 1998.

[35] T. Halpin, “Object Role Modeling - The Official Site for Conceptual Data Modeling.” available from http://www.orm.net/, 2011.

[36] B. Piprani, “Using ORM-Based Models as a Foundation for a Data Quality Firewall in an Advanced Generation Data Warehouse (Extended Version),” *J. Data Semant. XI*, pp. 94–125, 2008.

[37] J. Hansen and N. Dela Cruz, “Evolution of a dynamic multidimensional denormalization meta model using object role modeling,” *Move to Meaningful Internet Syst. 2006 …*, pp. 1160–1169, 2006.

[38] K. Evans, “Requirements engineering with ORM,” *Move to Meaningful Internet Syst. 2005 …*, vol. vol 3762, pp. 646–655, 2005.

[39] E. J. Pierson, N. Cruz, H. A. N, and S. Paul, “Using Object Role Modeling for Effective In-House Decision Support Systems 2 What It Takes to Create Guidant DSS Solutions,” pp. 636–645, 2005.

[40] T. Halpin, “Fact-Oriented Modeling : Past , Present and Future,” no. Xml.

[41] T. Halpin, “ORM 2 Graphical Notation,” no. September, pp. 1–17, 2005.

[42] H. M. Wagih, “Mapping Object Role Modeling 2 Schemes to OWL2 Ontologies,” 2005.

[43] T. Halpin, “Fact-Orientation and Conceptual Logic,” *2011 IEEE 15th Int. Enterp. Distrib. Object Comput. Conf.*, pp. 14–19, Aug. 2011.

[44] D. Dwelle, “InfoModeler acquired by Visio.” available from http://www.aisintl.com/case/products/infomodeler/infomdl.html#Conclusion, 2000.

[45] T. Halpin, “NORMA Tool.” availabl;e from http://www.orm.net, 2010.

[46] P. De Leenheer and C. Debruyne, “A Tool for Fact-Oriented Collaborative Ontology Evolution.”

[47] “UNIVERSITY OF OSLO Department of Informatics,” 2011.

[48] B. C., *ORMLite 13b download*. available from http://www.ormfoundation.org/files/folders/orm_lite/entry3147.aspx, 2012.

[49] “Edraw Max Pro.” available from http://www.edrawsoft.com/EDrawMax.php, 2011.

[50] M. Uschold and M. Gruninger, “Ontologies : Principles , Methods and Applications,” no. February, 1996.

[51] T. M. dos S. Gaspar, “Methodology for Collaborative Enterprise Reference Ontology Building,” 2011.

[52] J. P. Mccusker, J. Luciano, and D. L. Mcguinness, “Towards an Ontology for Conceptual Modeling.”

[53] J. Hebeler and A. Perez-lopez, *Semantic Web Programming*. Wiley, 2009.

[54] Larry Goldberg and B. Von Halle, “The Decision Model: A Live Primer.” available from http://openrules.com/docs/DecisionModelPrimer.htm#top, 2009.

[55] B. von Halle and L. Goldberg, *The Decision Model : A business Logic Framework Linking Business and Technology*. Taylor & Francis Group, LLC, 2010.

[56] “OpenRules: Decision Management System.” available at http://openrules.com/.

[57] I. S. O. Tc and I. E. C. Wd, “Information technology — Metamodel framework for interoperability (MFI) — Part xx : Part xx : Metamodel for information model registration – Fact Based Models,” no. 20, 2011.

[58] I. C. Society, *IEEE Standard Glossary of Software Engineering Terminology*. available from http://ieeexplore.ieee.org/servlet/opac?punumber=2238, 1990, pp. 1–84.

[59] L. Levine, “System of Systems Interoperability (SOSI):,” no. April, 2004.

[60] A. Gilchrist, “Thesauri, taxonomies and ontologies – an etymological note,” *J. Doc.*, vol. 59, no. 1, pp. 7–18, 2003.

[61] J. Park, “Information Systems Interoperability : What Lies Beneath ?,” vol. 22, no. 4, pp. 595–632, 2004.

[62] E. Yahia, M. Lezoche, A. Aubry, and H. Panetto, “Semantics enactment for interoperability assessment in Enterprise Information Systems Esma Yahia * , Mario Lezoche, Alexis Aubry, Hervé Panetto ,” vol. 1, pp. 101–117, 2012.

[63] A. Tolk, S. Y. Diallo, and C. D. Turnitsa, “Applying the Levels of Conceptual Interoperability Model in Support of Integratability , Interoperability , and Composability for System-of-Systems Engineering,” vol. 5, no. 5, pp. 65–74.

[64] J. Sarraipa, J. P. M. A. Silva, R. Jardim-gonçalves, and A. A. C. Monteiro, “MENTOR – A Methodology for Enterprise Reference Ontology Development,” 2008.

[65] C. Agostinho, J. Sarraipa, D. Gonçalves, and R. Jardim-Gonçalves, “Tuple-based semantic and structural mapping for a sustainable interoperability. Proceedings of: Technological Innovation for Sustainability,” *Second IFIP WG 5.5/SOCOLNET Dr. Conf. Comput. Electr. Ind. Syst. DoCEIS 2011, Costa Caparica, Port.*, 2011.

[66] S. R. G. Fraser, “Overview of the .NET Framework.” available at http://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.110).aspx, pp. 1–31, 2003.

[67] Microsoft, “.NET Framework.” available at http://msdn.microsoft.com/pt-br/vstudio/aa496123.

[68] R. Schumacher, “Dispelling Myths,” vol. 92, no. 18. available at http://web.archive.org/web/20110606013619/http://dev.mysql.com/tech-resources/articles/dispelling-the-myths.html, p. 1470a–1470, 20-Sep-2000.

[69] T. Nash, “LINQ: Language Integrated Query.” pp. 543–576, 2010.

[70] “An Introduction To RESTful Services With WCF.” - MSCD Magazine, http://msdn.microsoft.com/en-us/magazine/dd315413.aspx.

[71] J. Sarraipa, R. Jardim-Goncalves, and A. Steiger-Garcao, “MENTOR: an enabler for interoperable intelligent systems,” *Int. J. Gen. Syst.*, vol. 39, no. 5, pp. 557–573, Jul. 2010.